HOMOTOPY TYPE THEORY

Egbert Rijke

July 6, 2012

A MASTER THESIS FOR THE MATHEMATICAL SCIENCES PROGRAMME AT UTRECHT UNIVERSITY, THE NETHERLANDS.

SUPERVISORS: ANDREJ BAUER, BENNO VAN DEN BERG AND JAAP VAN OOSTEN.

WRITTEN IN LJUBLJANA, SLOVENIA.

CONTENTS

INTRODUCTION

In this thesis we will investigate type theory with identity types, which was invented by Per Martin-Löf in the 1970's. While the theory of identity types was originally intended to internalize the notion of equality in type theory, and thereby enableing one to prove equality of two terms by inhabiting the corresponding identity types, it has become apparent in the last decade that it provides a setting for doing formal homotopy theory. The main idea was to interpret a proof of equality for two terms a and b as a path from a to b. The pioneers of this interpretation were Steve Awodey, Michael Warren and Vladimir Voevodsky and the theory that resulted from their work is currently intensively investigated, both from it's type theoretical side and by providing new models for the theory. This thesis could be seen as a report on the recent type theoretical results in this program.

Chapter 1 In the first chapter we give a brief overview of (dependent) type theory. Type theory is not in the standard curricula of neither Utrecht University nor at the University of Ljubljana and therefore I felt that it was necessary to include a chapter of this sort. Although we keep the terms elementary and understandable for a mathematical audience, this chapter serves several aims. At several moments in the text we will point towards instances where it would be very helpful if we would have identity types and therefore directing the story to the parts of type theory that are for us the most relevant. Also, we give many examples of inductively defined types. The reader will see that many basic constructions in mathematics have their inductive counterparts in type theory. Understanding inductively defined types at a basic level is important for intensional type theory, since identity types are defined inductively. Moreover, in chapter 3 we will use the guiding principles of defining types inductively to study higher inductive types. As a teaser, we will prove the type theoretical analogue of the axiom of choice.

Chapter 2 In the second chapter the main part of the thesis begins. We will introduce identity types and develop the theory from the ground up. The results in this chapter originate from the Coq repositories, written by Mike Shulman, and from the original work of Vladimir Voevodsky. After having introduced identity types and the basic properties of paths, we will be mainly interested in the interaction between functions and paths. This will lead us to the theory of homotopies, the notion of contractability, equivalences of types and function extensionality. One of the more useful results about equivalences is that every isomorphism is an equivalence, which will simplify the search for equivalences between spaces significantly. Nevertheless, as soon as function spaces come into play, we will need the principle of function extensionality to complete any proof of equivalence. Function extensionality is not part of type theory and it needs to be assumed as an axiom. But we will see that the natural principle of function extensionality follows from a very weak form of function extensionality: the principle

that says that any product of contractible spaces is itself contractible. This principle could also be regarded as a weak version of the full form of the axiom of choice. In chapter 1 we introduced the axiom of choice as a certain function; the full form of the axiom of choice is that this function is an equivalence. While we derive the strong form of function extensionality from the weak, we will witness a delicate interplay between choice and extensionality.

After the exposition of the function extensionality principle and the axiom of choice, we investigate the similarities between paths in a type and equivalences between types. Every path between two types in a universe induces an equivalence between them. Voevodsky's univalence axiom then says that the transition from paths to equivalences is itself an equivalence. In loose terms: there are no other equivalences than those induced by paths. The univalence axiom is an extensionality principle for types. It gives us a way to construct a path between two types by constructing an equivalence between them. This idea can be taken a step further: we will give a proof that the univalence axiom implies function extensionality. After this is established, we give two new applications of the univalence axioms. The first is a correspondence between the space of dependent types over a type A and the space of functions to A. The second is a proof that a function from A to B is exactly what it would be in set theory: a total single-valued relation from A to B.

In the last part of this chapter we take a digression from the Coq repositories and prove a type theoretical analogue of the Yoneda lemma. Using the basic equivalences we had derived earlier, this is not a difficult task anymore. But it does justify a higher categorical point of view regarding types and dependent types over types. With the same techniques, we will be able to prove that various basic adjunctions of the theory of presheaves also translate to adjunctions between the spaces of dependent types. As far as I know, the results presented in this last part of chapter 2 are new, although I have heard from Andrej that some experts in the field have known them already before me.

Chapter 3 The second main chapter of this thesis deals with higher inductive types. We introduce higher inductive types by example. Higher inductive types are types that are defined inductively, but instead of only giving point constructors as it's basic constructors, we also allow ourselves to use path constructors as basic constructors. And of course higher paths... The first two examples we give are the interval, described as the inductive type with two points and a path between them as basic constructors, and the circle. Using function extensionality, we will be able to find correspondence principles that replace the induction principles we have given in both the cases of the interval and the circle. Those are equivalences in analogy to the bijections of homsets found for universal objects in category theory. Thus, we will see exactly *how* the induction principle is a universal property. Also, even though the interval turns out to be contractible, it's induction principle is the prototype for all higher inductive types with paths as basic constructors.

We then continue to develop the theory of higher paths to be able to formulate the induction principles for higher inductive types with higher paths as basic constructors. The two basic examples we give are the disc and the sphere. Once we have been able to formulate the induction principle for the disc, we will see that the induction principles

for all types with 2-paths as basic constructors are similar. Also, we will derive a correspondence theorem for both the disc and the sphere.

Conventions The fact that this is a thesis in constructive type theory also has its implication for the structure of the document. Lemmas and theorems, for instance, all define certain functions which are constructed in their proofs. Consequently, if an assertion is of the form 'there exists a term x in A such that…' will be proved by constructing an x with the asserted property. If we later make use of this assertion, we mean to make use of exactly that term x which has been constructed in the proof, contrary to the advise Serre had given in his talk[1] *'How to write mathematics badly'*. Likewise, the proof of an assertion of the form 'for all terms x of A we have…' is a function which takes terms of A as arguments. If we later make use of such an assertion, we mean to make use of the function we have constructed in its proof. These are not conventions which I have invented for the purpose of writing this thesis but it rather follows practice of how Coq code is written.

A remark on our notation for path spaces is also in place here. If p is a path from x to y, we denote this by $p : x \rightsquigarrow y$, following the suggestions of the earlier Coq repositories about homotopy type theory. The Coq notation has recently changed to $p : x == y$, which is more in line with the interpretation of p as a proof of propositional equality of x and y. In this thesis I have decided to stick with the former $\rightsquigarrow$ notation. One of the reasons is that we will work a lot with commutative diagrams, where the presentation is much clearer if the direction of the identity proofs is clear from the notation.

Shortcomings Originally, I had intended to concentrate more on interpretations of intensional type theory. Since Hofmann and Streicher had presented their groupoid model of intensional type theory, and thereby made it apparent that type theory is not extensional, various other models have been found. Among those, the most noticeable are the simplicial sets model and the interpretation of type theory in arbitrary model categories. The discovery of these models has contributed greatly to the momentum of this research area and it also raises the desire to find out what the various notions of type theory type theory become in these interpretations. I have made a very small step at this, by proving that the interpretation of the circle in the groupoid model is (equivalent to) the group of integers. But this leaves much to be desired: what is the circle in the model of simplicial sets, or in a model category of topological spaces? What about other higher inductive types: are they always the CW complexes you would guess they are, based on their basic constructors?

[1] The videos of his talk are available in three parts via the urls

part 1 http://www.dailymotion.com/video/xf88b5_jean-pierre-serre-writing-mathemati_tech

part 2 http://www.dailymotion.com/video/xf88en_jean-pierre-serre-writing-mathemati_tech

part 3 http://www.dailymotion.com/video/xf88g3_jean-pierre-serre-writing-mathemati_tech

Regrettably, I have not collected enough material on the interpretations of intensional type theory in the year that I have done the research for this thesis to fill a chapter. The observation that the groupoid interpretation of the circle is the integers is included in Appendix B.

A second point of concern is that I leave λ somewhat mysterious. I don't manage to explain well why it is that $\lambda x.f(x)$ is not provably equal to $\lambda x.g(x)$ if we have $f(x) = g(x)$ for all $x : A$. My current understanding of this is that λ actually takes entailments $x : A \vdash f(x) : P(x)$ as an argument, but I have never seen this written down somewhere.

A last shortcoming in this thesis is that I will not give much details about what universes are. It will be assumed that there is a hierarchy of universes, where a universe at level k is a type of the universe at level $k + 1$. I will mostly work in the lower few universes: at the bottom for types, one level higher for dependent types and occasionally another level up. But I must admit here that I haven't studied universes well enough to give a good explanation about them.

Acknowledgments I am very grateful for the guidance and insights Andrej Bauer gave me during this project. He has been very good at giving perspective while I was trying to learn and work with type theory. It happened several times that clumsy technical result of mine needed his conceptual understanding in order to be written down properly. I definitely owe my intuition in homotopy type theory to him.

I'd also like to thank Špela Špenko for her support. Chapter 1 of this thesis is written for her, so that she could get an idea of what I have been working on.

During the Fourth Workshop on Formal Topology I had many fruitful and insightful discussions. Some of them have even made it to this thesis. So I'd like to thank also Dan Licata and Bas Spitters.

Finally, I'd like to thank Benno van den Berg and Jaap van Oosten for reading an early version of this thesis and suggesting many improvements.

EGBERT RIJKE
JUNE 2012, LJUBLJANA

1. A SHORT GUIDE TO CONSTRUCTIVE TYPE THEORY

This short chapter is aimed at my fellow students and mathematicians who never had a course in type theory. The world of type theory is a bit like the world of set theory, in the sense that there are types A, B, C,... and those types can have terms $a_0, a_1, \ldots : A$. But types may have a hidden structure that keeps their terms in a shadowy corner where they are hard to get a hold on. We give status to this dark nature of the terms by allowing ourselves only the constructive methods to reason about terms. For example, to find a term of the product $A \times B$ of the types A and B the only thing we can do is constructing a term a of type A and a term b of type B and conclude that the pair $\langle a, b \rangle$ is a term of $A \times B$.

Initially, the vocabulary of type theory is rather small. In non-dependent type theory there are four things we can say, i.e. there are four judgments that can be made:

A is a type:	$A : \mathsf{Type}$
a is a term of type A:	$a : A$
A and B are equal types:	$A = B : \mathsf{Type}$
a and b are equal terms of type A:	$a = b : A$.

Dependent type theory, on which we will rely in this thesis, has furthermore the judgments

$P(a)$ is a type for every $a : A$:	$a : A \vdash P(a) : \mathsf{Type}$
$f(a) : P(a)$ for every $a : A$:	$a : A \vdash f(a) : P(a)$.

These can be thought of in terms of indexed sets, but in type theory they become indexed types or rather dependent types. In this work you might even catch us on using the term 'fibration' for dependent type, because dependent types are interpreted by fibrations when we interpret types as topological spaces. But for now, the indexed set interpretation will do and in that case, the judgment $a : A \vdash f(a) : P(a)$ is to be interpreted as a function that sends an element a of the indexing type A to the type $P(a)$. We call such a f a dependent function, or a section of the dependent type P. Recall from set theory that if $\{P_a : a \in A\}$ is an indexed set, then the set $\prod_{a \in A} P_a$ consists exactly of these sections. We will see a similar type $\prod(a : A),\ P(a)$ constructed in section 1.1.1.

The equality that we use here is 'equality in the strongest form', or definitional equality. The thing that you should keep in mind with definitional equality is that if f and g are functions from A to B such g has the same values as f but is computed in a different way, then the functions f and g are not definitionally equal even though we would consider them to be equal from a set theoretical point of view (by function extensionality). The point here is that we might have an interpretation of type theory in which there is a type containing two functions for which the number of steps a

function needs to compute the value at a certain point might be relevant, so we consider functions that involve different computations as different elements even if there is a proof that their values are always the same. The main topic of this thesis is the *propositional equality* which exactly addresses this point. The key distinction between definitional and propositional equality is that definitional equality is a syntactical notion while propositional equality is internal in the theory of types: there are types $\mathsf{Id}_A(x,y)$ of propositional equality for every two terms x and y of a given type A and when they are inhabited, i.e. when an element of $\mathsf{Id}_A(x,y)$ can be constructed, we say that x and y are propositionally equal. These types of propositional equality are the identity types of the next chapter 2.1. In this guide we will point to several instances where the presence of identity types is desired.

Nevertheless, there are a few things we can say about definitional equality. If a is a term of type A and $A = B : \mathsf{Type}$ then a is a term of type B. If P is a dependent type over A, i.e. if we have $a : A \vdash P(a) : \mathsf{Type}$, and if $a = b : A$, then we have $P(a) = P(b) : \mathsf{Type}$. These essentially say that we might freely substitute equal terms or types.

A small note: we will not use the $\vdash$ notation in this thesis and the notation with inference lines make no appearance at all. Instead, we have decided to stay syntactically closer to the notation in ordinary mathematics and the mathematical models that interpret type theory, so that readers from outside this area do not also have to bridge the linguistic gap.

1.1 A dependent type over a type

Suppose that A is a type. A 'function' that assigns to each term a of A a type $P(a)$ is said to be a dependent type over A. Dependent types are abundant in mathematics. Indexed sets form a main example, or indexed objects in a category (i.e. a functor from a set to a category). A continuous function $f : X \to A$ also gives rise to the dependent space $a \mapsto f^{-1}(\{a\})$ sending a point to its inverse image.

The notion of dependent type can be seen as a way to associate a property to the terms. Thus, if P is a dependent type over A, then an inhabitant u of the type $P(a)$ is a witness that this property holds for a. In that sense, a dependent type is also a way to talk about a 'subtype' of a type. A real analogue of the notion of subset in set theory does not exist in type theory because the association of the type A with a term a of type A is inherent in the sense that if a is a term of type A then it is not a term of another type B. Thus one can take the point of view that a dependent type over A is a way around this – from a set theoretical point of view – drawback.

Dependent types can also depend over other dependent types. If P is a dependent type over A, then we may have that $Q(a,u)$ is a type for every $a : A$ and $u : P(a)$. In that case, Q is said to be a dependent type over P. This generalizes of course to any number: we may have types depending on Q and so forth. All of these are called dependent types.

The point of view we take throughout this thesis is that a dependent type is a map of a type into a universe. We assume that there is a hierarchy of universes Type_1, Type_2,...,Type_ω, where each Type_k is closed under the basic constructions such as dependent products and dependent sums. Moreover, Type_k is a term of Type_{k+1}. Other than this, we will not be very precise about what universes are. One reason is that

the author is not very familiar with them, another reason is that they form a topic of current research. Also, we will not use the index notation and simply write Type. Hence a dependent type P over A is denoted by $P : A \to \mathsf{Type}$, where the level k of Type is assumed to be such that $A : \mathsf{Type}_k$.

1.1.1 Dependent products

Suppose that $P : A \to \mathsf{Type}$ is a dependent type over A. Then we may form the dependent product $\prod(a : A),\ P(a)$, which is a type that behaves like a product in set theory or a limit in category which is taken over a set. The terms of $\prod(a : A),\ P(a)$ are functions that take a term a of A to a term of type $P(a)$ which we call sections of P. The dependent product $\prod(a : A),\ P(a)$ has the following property: each judgement $a : A \vdash g(a) : P(a)$ defines a term

$$\lambda a.g(a) : \prod(a : A),\ P(a).$$

Furthermore, if we have a section $f : \prod(a : A),\ P(a)$ and an element $a : A$, then we have a term

$$\mathsf{evaluate}(f, a) : P(a).$$

Of course, we denote $\mathsf{evaluate}(f, a)$ simply by $f(a)$ later on. The constructors λ and evaluate satisfy the following property: if we can construct from each term $a : A$ a term $g(a)$ of $P(a)$ and if x is a term of A, then

$$\mathsf{evaluate}(\lambda a.g(a), x) = g(x) \qquad (\beta\text{-rule})$$

and when $f : \prod(a : A),\ P(a)$ is a section of P we have

$$\lambda a.\mathsf{evaluate}(f, a) = f \qquad (\eta\text{-rule})$$

The β-rule asserts that $\lambda a.g(a)$ extends the choice of $g(a)$ for $a : A$ while the η-rule asserts that everything in $\prod(a : A),\ P(a)$ is indeed a function and serves therefore as a minimality principle for dependent product types.

It may happen, of course, that a dependent type is actually non-dependent, i.e. that $P(a) = B$ for all $a : A$. In that case, an element of $\prod(a : A),\ P(a)$ is just a function from A to B, and we write $A \to B$ for the type $\prod(a : A),\ P(a)$.

The construction of dependent product easily generalizes to the situation where Q a is a dependent type over a dependent type P over A. The dependent product type $\prod(a : A),\ (\prod(u : P(a)),\ Q(a, u))$ is denoted simply by $\prod(a : A)(u : P(a)),\ Q(a, u)$.

A note on notation: If $f : \prod(a : A)(u : P(a)),\ Q(a, u)$ is a dependent function it may become cumbersome to write $f(a, u)$ for the value of f at the points $a : A$, $u : P(a)$. We would much rather write $f(u)$, since for u to be known, we must know the term a of A for which u is a type of $P(a)$. In other words, if u is known, then a is known implicitly. To indicate that we wish to omit the implicitly known variables in a dependent product type (and in a dependent sum type as well, we'll soon come to those), we will write the type as

$$\prod\{a : A\}(u : P(a)),\ Q(a, u)$$

This is the same type; the curly brackets around a variable only indicate that reference to that variable is omitted in the notation of the value of a function at that point.

1.1.2 Dependent sums

Suppose that $P : A \to \mathsf{Type}$ is a dependent type over a type A. Then there is the type $\sum(a:A),\ P(a)$, which we call the dependent sum or the total space of P. It behaves much like a disjoint sum of sets or like a colimit in a category taken over a set (i.e. no morphisms to mess things up). For the dependent sum we introduce a function

$$\langle -,- \rangle : \prod(a:A),\ \big(P(a) \to \sum(x:A),\ P(x)\big)$$

which includes every $a : A$ and $u : P(a)$ as the pair $\langle a,u\rangle$ in $\sum(a:A),\ P(a)$. Thus, the dependent sum consists of pairs. But our way of saying that it consists only of pairs is via a minimality principle similar to the universal property of a colimit in category. The minimality principle for $\sum(a:A),\ P(a)$ is the property that whenever we have a dependent type

$$Q : \big(\sum(a:A),\ P(a)\big) \to \mathsf{Type}$$

over $\sum(a:A),\ P(a)$ and a dependent function

$$q : \prod(a:A)(u:P(a)),\ Q(\langle a,u\rangle)$$

assigning to each of the pairs $a : A$ and $u : P(a)$, a term of type $Q(\langle a,u\rangle)$, there is a section $\sigma(Q,q) : \prod(w : \sum(a:A),\ P(a)),\ Q(w)$ of Q which satisfies

$$\sigma(\langle a,u\rangle) = q(a,u)$$

for each $a : A$ and $u : P(a)$. The last equality expresses that σ is an extension of q via the inclusion $\langle -,-\rangle$. This property basically says that to give a section of Q it is enough to assign a point of $Q(\langle a,u\rangle)$ for each $a : A$ and $u : P(a)$ and it is in that sense that $\sum(a:A),\ P(a)$ consists only of the pairs $\langle a,u\rangle$. Indeed, it allows us to prove that:

Lemma 1.1.1. *For any dependent type P over a type A there are functions*

$$\mathsf{proj}_1 : \big(\sum(a:A), P(a)\big) \to A,$$
$$\mathsf{proj}_2 : \prod\big(w : \sum(a:A),\ P(a)\big),\ P(\mathsf{proj}_1(w)).$$

with the property that $\langle a,u\rangle = \langle \mathsf{proj}_1\langle a,u\rangle, \mathsf{proj}_2\langle a,u\rangle\rangle$ for every $a : A$ and $u : P(a)$.

PROOF. The first projection is defined by letting $Q : (\sum(a:A),\ P(a)) \to \mathsf{Type}$ be the constant dependent type given by $Q(w) = A$, and by letting q be the function $\lambda(a,u).a$. The minimality principle then gives a function $\sigma : (\sum(a:A),\ P(a)) \to A$ with the property that $\sigma(\langle a,u\rangle) = a$. We define proj_1 to be this σ.

For the second projection we let $Q : (\sum(a:A),\ P(a)) \to \mathsf{Type}$ be the dependent type given by

$$Q(w) := P(\mathsf{proj}_1(w))$$

and let $q : \prod(a:A)(u:P(a)),\ Q(\langle a,u\rangle)$ be the map $\lambda(a,u).u$. Then the minimality principle gives a function $\sigma : \prod(w:\sum(a:A),\ P(a)),\ P(\mathsf{proj}_1(w))$ with the property that $\sigma(\langle a,u\rangle) = u$. We define proj_2 to be this σ. ∎

It should be noted, though, that w and $\langle \mathsf{proj}_1(w), \mathsf{proj}_2(w)\rangle$ do not need to be definitionally equal terms of $\sum(a:A),\ P(a)$ for every w. Instead, they are propositionally equal, we will prove this fact in **??**.

Another way of looking at the minimality principle for dependent sums is via the equivalence

$$((\exists x \varphi) \to \psi) \leftrightarrow \forall x(\varphi \to \psi)$$

in logic, which holds whenever the variable x does not occur in ψ. The minimality principle of $\sum(a:A),\ P(a)$ ensures that we have an equivalence

$$\Big(\Big(\sum(a:A), P(a)\Big) \to B\Big) \cong \prod(a:A),\ (P(a) \to B)$$

for every type B. We have not introduced the notion of equivalence yet, but this should clearly involve functions back and forth that behave like each others inverses in a generalized sense. In fact, we can (and will demonstrate it in a moment) get quite far with proving this equality, and we will get stuck exactly at the moment where we need a type that expresses equality of certain terms: the type of propositional equality.

A function F from $(\sum(a:A),\ P(a)) \to B$ to $\prod(a:A),\ P(a) \to B$ is given by precomposition with ι. A function G in the other direction is given by the defining property of the dependent sum by taking Q to be the constant dependent type $Q(w) = B$ and by taking q to be the function to which we wish to apply G. It is then immediate that G is a section of F, i.e. that $F \circ G$ is the identity map on $\prod(a:A),\ P(a) \to B$. But the argument to show that it is also a retraction is done via the defining property of the dependent sum using the dependent type over $\sum(a:A),\ P(a)$ that asserts that $G(F(h))(w) = h(w)$ for some $h : (\sum(a:A),\ P(a)) \to B$ and $w : \sum(a:A),\ P(a)$. These are instances of the identity types that we will investigate thoroughly in the next chapter. Therefore, we will postpone the full proof of the equivalence until lemma 2.6.3 in section 2.4.1 about equivalences.

It should be noted that it is possible to introduce the dependent sums in a different way, more similar to how we introduced the dependent product types, by stating that for every $w : \sum(a:A),\ P(a)$ there are elements $\mathsf{proj}_1(w)$ of type A and $\mathsf{proj}_2(w) : P(\mathsf{proj}_1(w))$, with the property that

$$w = \langle \mathsf{proj}_1(w), \mathsf{proj}_2(w)\rangle.$$

In this definition the space $\sum(a:A),\ P(a)$ is presented directly as a space of pairs $\langle a,u\rangle$ where $a : A$ and $u : P(a)$ while in our original definition this was hidden in the minimality principle.

1.2 Defining types inductively

Many types in type theory can be defined by stating an induction principle for them. To define a type inductively we have to specify the basic terms of that type and then we

have to state that that type satisfies a certain minimality principle. A good illustrative example of this is the type $\mathbb{N}$ of natural numbers. It has a term zero and a successor function $\mathsf{S} : \mathbb{N} \to \mathbb{N}$. To show that some property P holds for the natural numbers it suffices to show that that property holds for zero and that whenever the property holds for a number $n : \mathbb{N}$, then it also holds for $\mathsf{S}(n)$. In type theory this means that whenever $P : \mathbb{N} \to \mathsf{Type}$ is a dependent type over $\mathbb{N}$, to show that $\prod(n : \mathbb{N}),\ P(n)$ holds it suffices to give a term of type $P(\mathsf{zero})$ and a term of type

$$\prod(n : \mathbb{N}),\ P(n) \to P(\mathsf{S}(n)).$$

Thus, the induction principle for $\mathbb{N}$ becomes a way to define sections of the dependent types P over $\mathbb{N}$. In full, the induction principle for $\mathbb{N}$ becomes: whenever we have

$$\begin{aligned} &p : P(\mathsf{zero}) \\ &f : \prod(n : \mathbb{N}),\ P(n) \to P(\mathsf{S}(n)) \end{aligned}$$

then there is a section $\sigma : \prod(n : \mathbb{N}),\ P(n)$ with the property that $\sigma(\mathsf{zero}) = p$ and $f(n)(\sigma(n)) = \sigma(\mathsf{S}(n))$ for all $n : \mathbb{N}$. In this style we can define many types.

Example 1.2.1. We can define the type unit inductively by specifying one constructor $\mathsf{tt} : \mathsf{unit}$. The induction principle for unit becomes that whenever $P : \mathsf{unit} \to \mathsf{Type}$ is a dependent type over unit and $p : P(\mathsf{tt})$, then there is a section $\sigma : \prod(x : \mathsf{unit}),\ P(x)$ with the property that $\sigma(\mathsf{tt}) = p$. Using the dependent type of identity types that assert that x equals tt we will be able to show in lemma **??** that every element of unit is equal to tt. ⋆

Example 1.2.2. The empty type $\varnothing$ is defined to have no constructors and the induction principle is that whenever P is a dependent type over $\varnothing$, there exists a section of P. ⋆

Example 1.2.3. Suppose that A and B are types. Then we may define the type $A + B$ inductively to be the type with basic constructors

$$\begin{aligned} &\iota_A : A \to A + B \\ &\iota_B : B \to A + B. \end{aligned}$$

The induction principle for $A + B$ becomes that whenever we have a dependent type P over $A + B$ and if we have the functions

$$\begin{aligned} &k_A : \prod(a : A),\ P(\iota_A(a)) \\ &k_B : \prod(b : B),\ P(\iota_B(b)) \end{aligned}$$

then there exists a section $\sigma : \prod(w : A + B),\ P(w)$. This induction principle closely follows the logic principle

$$(\varphi \to \chi) \to (\psi \to \chi) \to (\varphi \vee \psi \to \chi).$$

to introduce a disjunction on the left. ⋆

Example 1.2.4. Suppose that A and B are types. Then we may define the type $A \times B$ inductively to be the type with basic constructors

$$\rho : A \to B \to A \times B$$

The induction principle for $A \times B$ becomes that whenever P is a dependent type over $A \times B$ for which there is a function

$$k : \prod (a : A)(b : B),\ P(\rho(a,b)),$$

then there is a section $\sigma : \prod (w : A \times B),\ P(w)$ with the property that $\sigma(\rho(a,b)) = k(a,b)$. Again, there is similarity between this induction principle and the principle

$$(\varphi \to \psi \to \chi) \to (\varphi \wedge \psi \to \chi)$$

to introduce a conjunction on the left. ⋆

Example 1.2.5. In fact, we have already seen an example of a type defined inductively: the dependent sum. When P is a dependent type over a type A, the dependent sum $\sum (a : A),\ P(a)$ was defined to have the basic constructor $\iota : \prod (a : A),\ P(a) \to \mathsf{Type}$ and the induction principle for $\sum (a : A),\ P(a)$ was that, whenever Q is a dependent type over $\sum (a : A),\ P(a)$, to give a section $\prod (w : \sum (a : A),\ P(a)),\ Q(w)$ it suffices to define it on the pairs $\langle a, u \rangle$ of terms $a : A$ and $u : P(a)$. We called this a minimality principle there, but it was in fact an induction principle. ⋆

The definition of inductive types is somewhat complicated, but the idea is that one can define a type inductively by specifying its basic constructors – in the case of the natural numbers those are zero and S – and a term of the type that asserts that there is a section of every dependent type over the type we are defining which admits the same structure of the basic constructors. With some experience it is possible to just guess the right induction principle for a type when its basic constructors are given.

The important type in this text, the identity types, also have an inductive definition and we will define many more types inductively. But instead of only considering functions with codomain A as basic constructors of a type A which we are defining inductively – this is the normal state of affairs in inductive type theory – we will allow ourselves basic constructors which specify elements in the identity types as well. This allows us to state that some elements, while they are obtained in different ways, are equal. For example, we might want to state that $\mathsf{S}(\mathsf{S}(\mathsf{zero}))$ is equal to zero and then we get a 'two-element set' instead of the whole set of natural numbers. How to define types in this way is one of the main themes of this text.

2. TYPE THEORY WITH IDENTITY TYPES

We start here our investigation of type theory with identity types, which is also known as intensional type theory. If A is a type and x and y are terms of that type, then the identity type $\mathsf{Id}_A(x,y)$ can be thought of as the type that asserts that x and y are equal terms of A, or terms with equal values. The terms of $\mathsf{Id}_A(x,y)$ can then be treated as reasons why x and y are equal.

But Awody, Warren and Voevodsky have taught us that it is fruitful to think of the type $\mathsf{Id}_A(x,y)$ as the type of paths from x to y. We will see that there will emerge a whole homotopy theory in the theory of types, just as a consequence of assuming identity types. Moreover, intensional type theory has been interpreted in the model categories of ω-groupoids, simplicial sets and many others, where a dependent type over a type became a fibration over a space. In fact, the groupoid interpretation was the first interpretation of intensional type theory which had non-trivial paths. This was a noticeable achievement, since the only constructors of identity types are the reflexivity terms $\mathsf{id}_A : \mathsf{Id}_A(a,a)$ – the canonical proofs that a is equal to itself.

Identity types force us to think of types in a different way than as a kind of sets. Encouraged by the path interpretation of the terms of the identity types, we shall use the words 'type' and 'space' interchangeably and the terms of the spaces $\mathsf{Id}_A(x,y)$ shall always be called paths. It shall soon become clear that we have the same operations on paths as we are used to in topology: there is a notion of composition, we can invert paths, if P is a dependent space over A then paths in A can be lifted to the space $\sum(a:A),\ P(a)$ and if $f : A \to B$ is a function then f acts functorially on the paths of A. Note that the fact that functions act on paths tells us that all functions in type theory are continuous.

Paths give us the notion of equivalence, a way of saying that two types are the same that is much less restrictive that a notion of isomorphism one might think of. Indeed, we will see that any path between types induces an equivalence between those types. Voevodsky took this a step further by reasoning that the spaces of paths between types and the spaces of equivalences between types are themselves equivalent. This is his univalence axiom, which allows us to interchange equivalent spaces in an argument without any compromises. The univalence axiom has far reaching consequences: one of the first things Voevodsky did with univalence was showing that he got function extensionality as a consequence. Univalence has also been used to show that the fundamental group of the circle, which we will be able to define using identity types, is the group of integers. Moreover, it has been proven that the univalence axiom holds in the simplicial set interpretation of intensional type theory.

2.1 The inductive definition of identity types

We begin by introducing identity types and the basic notions that follow from them. For any type A, the identity type Id_A is of the type $A \to A \to \mathsf{Type}$. Thus, we have types

$\mathsf{Id}_A(x,y)$ for any $x,y:A$. One of the earliest interpretations of the types $\mathsf{Id}_A(x,y)$ is that of the propositional equality of x and y, i.e. of proofs that x and y are equal terms of type A. It is convenient, however, to think of inhabitants of $\mathsf{Id}_A(x,y)$ as paths from x to y. Like paths, they can be composed, inverted and they can be lifted, referring to the path lifting property in the theory of fibrations (in topological spaces).

Definition 2.1.1. The identity type $\mathsf{Id}_A : A \to A \to \mathsf{Type}$ for a space A is defined inductively with the dependent function

$$\mathsf{id} : \prod(a:A),\ \mathsf{Id}_A(a,a).$$

as its only basic constructor. The terms id_a point to the canonical paths from a to a, which are the constant paths in the topological interpretation. Note, however, that this inductive definition is unlike the inductively defined spaces we have seen so far, in the sense that Id_A is a dependent type over $A \times A$. Nevertheless, the induction principle for Id_A is very similar to the induction principles in section 1.2.

The induction principle for identity types says that to define a section for a dependent type $D : \prod\{x,y:A\},\ \mathsf{Id}_A(x,y) \to \mathsf{Type}$ over Id_A, it is enough to give a function $d : \prod(a:A),\ D(\mathsf{id}_a)$. Thus, whenever there are

$$\begin{aligned} D &: \prod\{x,y:A\},\ \mathsf{Id}_A(x,y) \to \mathsf{Type} \\ d &: \prod(a:A),\ D(a,a,\mathsf{id}_a) \end{aligned}$$

there is a section $J(D,d) : \prod\{x,y:A\}(p:\mathsf{Id}_A(x,y)),\ D(x,y,p)$ with the property that

$$J(D,d,\mathsf{id}_a) = d(a) \qquad \text{(conversion rule)}$$

holds for every $a:A$, stating that $J(D,d)$ extends the function d in the appropriate sense. As we have indicated by the curly bracket notation, we will usually write $J(D,d,p)$ instead of $J(D,d,x,y,p)$.

We will call the terms of $\mathsf{Id}_A(x,y)$ paths and we will denote the type $\mathsf{Id}_A(x,y)$ by $x \rightsquigarrow y$ accordingly. Thus, the identity type $\mathsf{Id}_A : A \to A \to \mathsf{Type}$ is also called the path space of A. ♦

As a first application of the induction principle for paths, we can use the induction principle for identity types to invert terms of type $x \rightsquigarrow y$ to obtain terms of $y \rightsquigarrow x$.

Lemma 2.1.2. *For every type A and every two terms $x,y:A$ there is a function*

$$\mathsf{inv} : (x \rightsquigarrow y) \to (x \rightsquigarrow y)$$

with the property that $\mathsf{inv}(\mathsf{id}_x) = \mathsf{id}_x$ for each $x:A$. We will usually denote $\mathsf{inv}(p)$ by p^{-1}.

PROOF. Let $D : \prod(x,y:A),\ (x \rightsquigarrow y) \to \mathsf{Type}$ be the dependent type over Id_A given by $D(x,y,p) :\equiv y \rightsquigarrow x$. Then we have the term

$$d :\equiv \lambda x.\mathsf{id}_x : \prod(x:A),\ D(x,x,\mathsf{id}_x)$$

and hence J gives us a term $J(D,d,x,y,p) : y \rightsquigarrow x$ for each $p : x \rightsquigarrow y$. Thus, $\mathsf{inv} := J(D,d,x,y)$ is a function from $x \rightsquigarrow y$ to $y \rightsquigarrow x$. The conversion rule gives the equality $\mathsf{inv}(\mathsf{id}_x) = \mathsf{id}_x$. ∎

Similarly, we may use the induction principle to compose paths:

Lemma 2.1.3. *For every type A and every three terms $x,y,z : A$ there is a function*

$$- \bullet - : (y \rightsquigarrow z) \to (x \rightsquigarrow y) \to (x \rightsquigarrow z)$$

giving the composition $q \bullet p : x \rightsquigarrow z$ for any pair of paths $q : y \rightsquigarrow z$ and $p : x \rightsquigarrow y$. This composition satisfies $\mathsf{id}_y \circ p = p$ for each $p : x \rightsquigarrow y$.

PROOF. Let D be the dependent type over Id_A given by

$$D(y,z,q) := (x \rightsquigarrow y) \to (x \rightsquigarrow z).$$

Then we have the dependent function

$$d := \lambda y.\mathrm{idmap}_{x \rightsquigarrow y} : \prod (y : A),\ D(y,y,\mathsf{id}_y),$$

where $\mathrm{idmap}_{x \rightsquigarrow y}$ is the identity function on $x \rightsquigarrow y$ defined by $\lambda p.p$. Now we may apply the induction principle for identity types to conclude that there is a term $J(D,d,q) : (x \rightsquigarrow y) \to (x \rightsquigarrow z)$ for every path $q : y \rightsquigarrow z$ in A giving composition. The equality $\mathsf{id}_y \circ p = p$ follows from the conversion rule. ∎

The following lemma describes the basic behavior of the operations of inversion and composition, justifying the nomenclature. A noteworthy distinction with the behavior of inversion and composition in a category is that they are *up to paths one level higher*, i.e. paths of paths. Indeed, the types $x \rightsquigarrow y$ have path spaces of their own and we use these to show that the usual familiar properties hold for inversion and composition, with equalities replaced by paths.

Lemma 2.1.4 (The ω-groupoid structure of types). *Suppose A and B are types, that $x,y,z,a : A$ and that $p : x \rightsquigarrow y$, $q : y \rightsquigarrow z$ and $r : z \rightsquigarrow a$. We have the following:*

i. There exists a paths $\mathsf{reflRight} : p \rightsquigarrow p \bullet \mathsf{id}_x$.

ii. There are paths $\mathsf{invLeft} : p^{-1} \bullet p \rightsquigarrow \mathsf{id}_x$ *and* $\mathsf{invRight} : p \bullet p^{-1} \rightsquigarrow \mathsf{id}_y$.

iii. There is a path $\mathsf{invTwice} : (p^{-1})^{-1} \rightsquigarrow p$.

iv. There is a path $\mathsf{assoc} : r \bullet (q \bullet p) \rightsquigarrow (r \bullet q) \bullet p$.

PROOF. All the proofs use the induction principle for paths.

i. Let D be the dependent type over Id_A given by

$$D(x,y,p) := p \rightsquigarrow p \bullet \mathsf{id}_x.$$

Then $D(x,x,\mathsf{id}_x)$ is the space $\mathsf{id}_x \rightsquigarrow \mathsf{id}_x \bullet \mathsf{id}_x$. Since $\mathsf{id}_x \bullet \mathsf{id}_x = \mathsf{id}_x$, it follows that $D(x,x,\mathsf{id}_x) = \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$. Thus, there is a term

$$d := \lambda x.\mathsf{id}_{\mathsf{id}_x} : \prod(x:A),\ D(x,x,\mathsf{id}_x).$$

Now J gives a term $J(D,d,p) : p \rightsquigarrow p \bullet \mathsf{id}_x$ for each path $p : x \rightsquigarrow y$ in A.

ii. Let D be the dependent type over Id_A given by

$$D(x,y,p) := p^{-1} \bullet p \rightsquigarrow \mathsf{id}_x.$$

Then $D(x,x,\mathsf{id}_x)$ is the space ${\mathsf{id}_x}^{-1} \bullet \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$. Since ${\mathsf{id}_x}^{-1} = \mathsf{id}_x$ and since $\mathsf{id}_x \bullet \mathsf{id}_x = \mathsf{id}_x$, we get that $D(x,x,\mathsf{id}_x) = \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$. Hence we find the term

$$d := \lambda x.\mathsf{id}_{\mathsf{id}_x} : \prod(x:A),\ D(x,x,\mathsf{id}_x).$$

Now J gives a term $J(D,d,p) : p^{-1} \bullet p \rightsquigarrow \mathsf{id}_x$ for each path $p : x \rightsquigarrow y$ in A. A path from $p \bullet p^{-1} \rightsquigarrow \mathsf{id}_y$ is found using a similar argument.

iii. Let D be the dependent type over Id_A given by

$$D(x,y,p) := (p^{-1})^{-1} \rightsquigarrow p.$$

Then $D(x,x,\mathsf{id}_x)$ is the space $({\mathsf{id}_x}^{-1})^{-1} \rightsquigarrow \mathsf{id}_x$. Since ${\mathsf{id}_x}^{-1} = \mathsf{id}_x$ for each $x : A$, we see that $D(x,x,\mathsf{id}_x) = \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$. Hence we find the term

$$d := \lambda x.\mathsf{id}_{\mathsf{id}_x} : \prod(x:A),\ D(x,x,\mathsf{id}_x).$$

Now J gives a term $J(D,d,p) : (p^{-1})^{-1} \rightsquigarrow p$ for each $p : x \rightsquigarrow y$ in A.

iv. Let D be the dependent type over Id_A given by

$$D(z,a,r) := \prod(x,y:A)(p:x \rightsquigarrow y)(q:y \rightsquigarrow z),\ r \bullet (q \bullet p) \rightsquigarrow (r \bullet q) \bullet p.$$

Then $D(z,z,\mathsf{id}_z)$ is the space

$$\prod\{x,y:A\}(q:y \rightsquigarrow z)(p:x \rightsquigarrow y),\ \mathsf{id}_z \bullet (q \bullet p) \rightsquigarrow (\mathsf{id}_z \bullet q) \bullet p,$$

which simplifies to

$$\prod\{x,y:A\}(q:y \rightsquigarrow z)(p:x \rightsquigarrow y),\ q \bullet p \rightsquigarrow q \bullet p.$$

Therefore, we find the term

$$d := \lambda z.\lambda x,y,p,q.\mathsf{id}_{q \bullet p} : \prod(z:A),\ D(z,z,\mathsf{id}_z)$$

Now J gives us a term $J(D,d,r,q,p) : r \bullet (q \bullet p) \rightsquigarrow (r \bullet q) \bullet p$ for each path $r : z \rightsquigarrow a$ in A. ∎

The analogy with paths in topological spaces is very strong. Also in a topological space paths can be composed and inverted. But composing a path with it's own inverse only gives a constant path *up to homotopy*, i.e. up to a higher path. Here, the constant paths in a topological space play the role of the identity paths in a type. Likewise, associativity holds only up to homotopy. Our intuition with paths in type theory relies very much on these ideas.

Now we wish to establish that functions $f : A \to B$ behave functorial. More precisely, that functions carry paths from their domain to their codomain. Another way of viewing this is that every function in type theory is continuous.

Lemma 2.1.5. *Suppose that $f : A \to B$ is a function and that $p : x \rightsquigarrow y$ is a path in A. Then there is a path*

$$f'(p) : f(x) \rightsquigarrow f(y)$$

in B. Moreover, for each $x : A$ we have $f'(\mathsf{id}_x) = \mathsf{id}_{f(x)}$.

PROOF. Let D be the dependent type over Id_A given by $D(x,y,p) := f(x) \rightsquigarrow f(y)$. Then we have the term

$$d := \lambda x.\mathsf{id}_{f(x)} : \prod(x : A),\ D(x,x,\mathsf{id}_x).$$

Now J gives us a path $f'(p) := J(D,d,p) : f(x) \rightsquigarrow f(y)$ in B for each path $p : x \rightsquigarrow y$ in A. The conversion rule gives that $f'(\mathsf{id}_x) = \mathsf{id}_{f(x)}$ for each $x : A$. ∎

The $'$ in our notation indicates that this is a non-dependent version of something which also has a dependent version, but usually we will not denote the $'$. In lemma 2.1.7 we will see how dependent function $f : \prod(x : A),\ P(x)$ acts on paths. To get there, we need another property of paths which is of primary interest of itself.

Lemma 2.1.6 (Transport). *Suppose that P is a dependent type over A and that $p : x \rightsquigarrow y$ is a path in A. Then there is a function $\mathsf{transport}(p) : P(x) \to P(y)$. For simplicity, we denote the term $\mathsf{transport}(p)(u) : P(y)$ by $p \cdot u$, for terms $u : P(x)$.*

PROOF. The dependent type $D : \prod(x,y : A),\ \mathsf{Id}_A(x,y) \to \mathsf{Type}$ we take is given by $D(x,y,p) := P(x) \to P(y)$. Then we have the function

$$d := \lambda x.\mathsf{id}_{P(x)} : \prod(x : A),\ D(x,x,\mathsf{id}_x),$$

so the induction principle for identity types gives us terms $J(D,d,p) : P(x) \to P(y)$ for $p : x \rightsquigarrow y$, which we define to be our $\mathsf{transport}(p)$. ∎

Another view on the transportation lemma is via propositional equality. Recall that a dependent type P over a type A can be seen as a property, varying over the terms of A. The property P holds for x in A if $P(x)$ is inhabited. Then the transportation lemma says that if x is propositionally equal to y, then $P(x)$ is inhabited if and only if $P(y)$ is inhabited. In the section about equivalences we will see that $P(x)$ and $P(y)$ are equivalent types if x and y are propositionally equal.

Lemma 2.1.7 (Dependent map). *Suppose that $f : \prod(x : A),\ P(x)$ and that $p : x \rightsquigarrow y$ is a path in A. Then there is a path $f(p) : p \cdot f(x) \rightsquigarrow f(y)$ in $P(y)$.*

PROOF. Let D be the dependent type over Id_A given by

$$D(x,y,p) :\equiv p \cdot f(x) \rightsquigarrow f(y).$$

Then $D(x,x,\mathsf{id}_x)$ is the space $\mathsf{id}_x \cdot f(x) \rightsquigarrow f(x)$. Since $\mathsf{id}_x \cdot f(x) = f(x)$, we get that $D(x,x,\mathsf{id}_x) = f(x) \rightsquigarrow f(x)$. Thus, we find the term

$$d :\equiv \lambda x.\mathsf{id}_{f(x)} : \prod(x:A),\ D(x,x,\mathsf{id}_x)$$

and now J gives us a path $J(D,d,p) : p \cdot f(x) \rightsquigarrow f(y)$ for each path $p : x \rightsquigarrow y$ in A. ∎

In the following lemma we prove that if $P : A \to \mathsf{Type}$ is a dependent type over A, then paths $p : x \rightsquigarrow y$ lift to paths in the total space $\sum(x:A),\ P(x)$. In other words, dependent types have the path lifting property. This means that we can think of a total space $\sum(x:A),\ P(x)$ as a *fibered space* over the space A, with the first projection being the fibration. The space $P(x)$ is then the fiber of $\sum(x:A),\ P(x)$ above x.

Lemma 2.1.8 (Path lifting property). *Suppose that P is a dependent type over A, that $p : x \rightsquigarrow y$ is a path in A and that $u : P(x)$. Then there is a path*

$$p_\Sigma(u) : \langle x,u\rangle \rightsquigarrow \langle y, p\cdot u\rangle$$

in the total space $\sum(x:A),\ P(x)$ of P.

PROOF. Let the dependent type D over Id_A be given by

$$D(x,y,p) :\equiv \prod(u:P(x)),\ \langle x,u\rangle \rightsquigarrow \langle y, p\cdot u\rangle.$$

Since $\mathsf{id}_x \cdot u = u$ for all $u : P(x)$ it follows that $D(x,x,\mathsf{id}_x) = \langle x,u\rangle \rightsquigarrow \langle x,u\rangle$, and hence we have the dependent function

$$d :\equiv \lambda x.\lambda u.\mathsf{id}_{\langle x,u\rangle} : \prod(x:A),\ D(x,x,\mathsf{id}_x).$$

The induction principle for identity types now gives us that there is a path $\langle x,u\rangle \rightsquigarrow \langle y, p\cdot u\rangle$ in $\sum(x:A),\ P(x)$ for each path $p : x \rightsquigarrow y$ in A, which we define to be our $p_\Sigma(u)$. ∎

The path lifting property may be used to find a path $\langle x, f(x)\rangle \rightsquigarrow \langle y, f(y)\rangle$ in $\sum(x:A),\ P(x)$ for every section f of P.

Corollary 2.1.9. *If $f : \prod(x:A),\ P(x)$ is a section of P and $p : x \rightsquigarrow y$ is a path in A, then there is a path from $\langle x, f(x)\rangle$ to $\langle y, f(y)\rangle$ in the total space $\sum(x:A),\ P(x)$.*

PROOF. By the path lifting property, we have a path

$$p_\Sigma(f(x)) : \langle x, f(x)\rangle \rightsquigarrow \langle y, p\cdot f(x)\rangle$$

for each path $p : x \rightsquigarrow y$ in A. Therefore, it suffices to find a path $\langle y, p \cdot f(x)\rangle \rightsquigarrow \langle y, f(y)\rangle$. Recall from the dependent map lemma that we already have a path $f(p) : p \cdot f(x) \rightsquigarrow f(y)$ in $P(y)$. Now note that the function

$$\langle y, -\rangle : P(y) \to \sum(x : A),\ P(x)$$

also acts on paths, i.e. we have a path $\langle y, f(p)\rangle : \langle y, p \cdot f(x)\rangle \rightsquigarrow \langle y, f(y)\rangle$ in the total space $\sum(x : A),\ P(x)$. ∎

In the last part of this first section about identity types we will show how we can use identity types to define functions

$$\begin{aligned}&\mathsf{proj}_1 : \Big(\sum(x : A),\ P(x)\Big) \to A\\ &\mathsf{proj}_2 : \prod\Big(w : \sum(x : A),\ P(x)\Big),\ P(\mathsf{proj}_1(w)).\end{aligned}$$

for each dependent type P over a type A. Moreover, there is a path

$$\langle \mathsf{proj}_1\, w, \mathsf{proj}_2\, w\rangle \rightsquigarrow w$$

for each $w : \sum(x : A),\ P(x)$.

Lemma 2.1.10. *Suppose that P is a dependent space over a space A. Then for every element w of $\sum(x : A),\ P(x)$ there exists a term $x : A$ and a term $u : P(x)$ for which there is a path from w to $\langle x, u\rangle$. We denote x by $\mathsf{proj}_1\, w$ and we denote u by $\mathsf{proj}_2\, w$.*

PROOF. Consider the dependent type Q over $\sum(a : A),\ P(a)$ given by

$$Q(w) = \sum(a : A)(u : P(a)),\ w \rightsquigarrow \langle a, u\rangle$$

and define the dependent function $K : \prod(a : A)(u : P(a)),\ Q(\langle a, u\rangle)$ to be given by

$$\lambda a, u.\langle a, u, \mathsf{id}_{\langle a,u\rangle}\rangle.$$

The induction principle for $\sum(a : A),\ P(a)$ then gives us a section of Q, which proves the lemma. ∎

Note that the conversion rule for the induction principle for dependent sums gives that $\mathsf{proj}_1\langle x, u\rangle = x$ and that $\mathsf{proj}_2\langle x, u\rangle = u$ for each $x : A$ and $u : P(x)$.

In this section we have given all the proofs with path induction in full detail. Since proofs with path induction are usually straight forward, future proofs will be given by only indicating that the proof is by path induction.

2.2 More properties of paths

In this section we investigate more properties of paths, of the transport function and of dependent maps. The lemmas listed here will be used to prove assertions later on, and

as such this section is here mainly for reference. They are mostly about interactions of transportation and (dependent) functions with composition and inversion. We will also cover some basic properties of higher paths. The only new definition in this section, which will be used extensively in this paper, is that of a commutative square of paths.

2.2.1 Preservation of composition

We now present a series of lemmas about how transportation and dependent functions interact with composition.

Lemma 2.2.1. *Suppose that P is a dependent type over A. For every two paths $p : x \rightsquigarrow y$ and $q : y \rightsquigarrow z$ in A there is a function*

$$\mathsf{tc}(q,p) : \prod(u : P(x)),\ (q \bullet p) \cdot u \rightsquigarrow q \cdot (p \cdot u),$$

asserting that $(q \bullet p) \cdot u$ and $q \cdot (p \cdot u)$ are propositionally equal for every $u : P(x)$.

PROOF. Let the dependent type D over Id_A be given by

$$D(y,z,q) :\equiv \prod(x : A)(p : x \rightsquigarrow y)(u : P(x)),\ (q \bullet p) \cdot u \rightsquigarrow q \cdot (p \cdot u).$$

To apply path induction, we need to find a term of type $D(y,y,\mathsf{id}_y)$ for each $y : A$. Note that we have $\mathsf{id}_y \cdot (p \cdot u) = p \cdot u$ and we have that $\mathsf{id}_y \bullet p = p$. Therefore, we have that

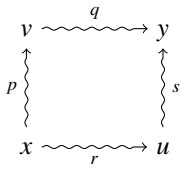
$$D(y,y,\mathsf{id}_y) = \prod(x : A)(p : x \rightsquigarrow y)(u : P(x)),\ p \cdot u \rightsquigarrow p \cdot u,$$

which is inhabited by the function

$$d :\equiv \lambda y.\lambda x, p, u.\mathsf{id}_{p \cdot u}$$

Now, the induction principle for paths gives us a term $\mathsf{tc}(q)$ of $D(y,z,q)$ for each path $q : y \rightsquigarrow z$ in A. ∎

Definition 2.2.2. Suppose that in a type A, we have paths as indicated in the diagram



We say that the above diagram commutes if there is a path $q \bullet p \rightsquigarrow s \bullet r$ in the space $x \rightsquigarrow y$. A path $h : q \bullet p \rightsquigarrow s \bullet r$ is called a witness of the commutativity of the diagram. Likewise, we may speak of a commutative triangle or a commutative diagram. In this text, a commutative diagram is always understood to commute up to propositional equality, unless stated otherwise. ♦

Lemma 2.2.3. *Suppose that P is a dependent type over A and that $p : x \rightsquigarrow y$ and $q : y \rightsquigarrow z$ are paths in A. Then the square*

$$\begin{array}{ccc} \langle y, p\cdot u\rangle & \overset{q_\Sigma(p\cdot u)}{\rightsquigarrow} & \langle z, q\cdot(p\cdot u)\rangle \\ {\scriptstyle p_\Sigma(u)}\uparrow & & \uparrow{\scriptstyle \langle z,\mathsf{tc}(q,p,u)\rangle} \\ \langle x, u\rangle & \underset{(q\bullet p)_\Sigma(u)}{\rightsquigarrow} & \langle z, (q\bullet p)\cdot u\rangle \end{array}$$

of paths in the total space $\sum(x:A),\ P(x)$ commutes for every $u : P(x)$.

PROOF. Let D be the dependent type over Id_A, with $D(y,z,q)$ defined by

$$\prod(x:A)(p:x\rightsquigarrow y)(u:P(x)),\ q_\Sigma(p\cdot u)\bullet p_\Sigma(u) \rightsquigarrow \langle z,\mathsf{tc}(p,q,u)\rangle\bullet(q\bullet p)_\Sigma(u)$$

For the identity path id_y on y, we have that $\mathsf{tc}(p,\mathsf{id}_y,u) = \mathsf{id}_{p\cdot u}$ and we have that $\mathsf{id}_{y\Sigma}(p\cdot u) = \mathsf{id}_{\langle y,p\cdot u\rangle}$. Also, note that $\langle y,\mathsf{id}_{p\cdot u}\rangle = \mathsf{id}_{\langle y,p\cdot u}$. So $D(y,y,\mathsf{id}_y)$ is the space

$$\prod(x:A)(p:x\rightsquigarrow y)(u:P(x)),\ p_\Sigma(u)\rightsquigarrow p_\Sigma(u),$$

which is inhabited by the function $\lambda x,p,u.\mathsf{id}_{p_\Sigma(u)}$. Thus we may conclude with the path induction principle that the asserted square commutes. ∎

Lemma 2.2.4. *Suppose that P is a dependent type over A and that $f : \prod(x:A),\ P(x)$ is a section of P. Then the square*

$$\begin{array}{ccc} f(z) & \overset{f(q)}{\leftsquigarrow} & q\cdot f(y) \\ {\scriptstyle f(q\bullet p)}\uparrow & & \uparrow{\scriptstyle q\cdot f(p)} \\ (q\bullet p)\cdot f(x) & \underset{\mathsf{tc}(q,p,f(x))}{\rightsquigarrow} & q\cdot(p\cdot f(x)) \end{array}$$

of paths in $P(z)$ commutes. Here $q\cdot f(p)$ denotes that the function $\mathsf{transport}(q) : P(y)\to P(z)$ is applied to the path $f(p)$.

PROOF. We will give the induction argument in a different way this time. Filling in id_y for q in the above diagram, we get that

$$\begin{array}{ccc} f(y) & \overset{f(\mathsf{id}_y)}{\leftsquigarrow} & \mathsf{id}_y\cdot f(y) \\ {\scriptstyle f(\mathsf{id}_y\bullet p)}\uparrow & & \uparrow{\scriptstyle \mathsf{id}_y\cdot f(p)} \\ (\mathsf{id}_y\bullet p)\cdot f(x) & \underset{\mathsf{tc}(\mathsf{id}_y,p,f(x))}{\rightsquigarrow} & \mathsf{id}_y\cdot(p\cdot f(x)) \end{array}$$

Converting all the terms involving id_y, we see that we have to provide a path from $f(p)$ to $f(p)$, which we take to be $\mathsf{id}_{f(p)}$. ∎

2.2.2 Preservation of inversion

Note the symmetry in the situation of the following lemma. This means that the function we assert to exist, exists in both directions.

Lemma 2.2.5. *Suppose that P is a dependent type over A. For every path $p : x \rightsquigarrow y$ there is a function*

$$\mathsf{mvTransp}(p) : \prod\{u : P(x)\}\{v : P(y)\},\ (p \cdot u \rightsquigarrow v) \to (u \rightsquigarrow p^{-1} \cdot v).$$

PROOF. Immediate with path induction over p. ∎

In fact, we can show that instead of the function space $(p \cdot u \rightsquigarrow v) \to (u \rightsquigarrow p^{-1} \cdot v)$ we can take a path space. In section 2.4 we will then be able to show that $p \cdot u \rightsquigarrow v$ and $u \rightsquigarrow p^{-1} \cdot v$ are equivalent spaces.

Lemma 2.2.6. *Suppose that P is a dependent type over A. For every path $p : x \rightsquigarrow y$ there is a function*

$$\prod(u : P(x))(v : P(y)),\ (p \cdot u \rightsquigarrow v) \rightsquigarrow (u \rightsquigarrow p^{-1} \cdot v).$$

PROOF. Immediate with path induction over p. ∎

In the following lemma we provide a way to compute $f(p^{-1})$ in terms of $f(p)$.

Lemma 2.2.7. *For every section f of a dependent space P over a space A and every path $p : x \rightsquigarrow y$ in A, there is a path*

$$f(p^{-1}) \rightsquigarrow \mathsf{mvTransp}(p, f(p))^{-1}.$$

PROOF. Immediate with induction on p. ∎

2.2.3 The dependent type $\mathcal{Y}(a)$

We introduce there the dependent type $\mathcal{Y}(a)$ over A, which will play the key role in section 2.8 about a type theoretical Yoneda lemma. Here we will use it to show a property of paths which will eventually enable us to prove contractibility of the spaces $\sum(x : A),\ x \rightsquigarrow a$ for each $a : A$ and of the interval, which we define in chapter 3.

Definition 2.2.8. Suppose that A is a type and that $a : A$. Define the dependent type $\mathcal{Y}(a)$ over A by

$$\mathcal{Y}(a)(x) :\equiv x \rightsquigarrow a$$

for each $x : A$. ♦

Lemma 2.2.9. *For any path $p : x \rightsquigarrow y$ in a space A there is a path from $p \cdot p$ to id_y, where the transportation is taken with respect to $\mathcal{Y}(y)$.*

PROOF. For a path $p : x \rightsquigarrow y$ let $D(x,y,p)$ be the space

$$\mathsf{transport}(P,p,p) \rightsquigarrow \mathsf{id}_y$$

Then $D(y,y,\mathsf{id}_y)$ is the space $\mathsf{id}_y \rightsquigarrow \mathsf{id}_y$, which contains the canonical term $d(y) = \mathsf{id}_{\mathsf{id}_y}$. Thus by the induction principle for identity types, there is a path from $p \cdot p$ to id_y for any path p of A. ∎

This lemma has a converse. Take $\mathcal{Y}^!(a)$ to be the dependent type over A given by $\mathcal{Y}^!(a)(x) :\equiv a \rightsquigarrow x$.

Lemma 2.2.10. *For any path $p : x \rightsquigarrow y$ in A there is a path from $p^{-1} \cdot p$ to id_x, where the transportation is taken with respect to $\mathcal{Y}^!(x)$.*

2.2.4 Higher paths

Lemma 2.2.11. *Suppose that P is a dependent type over a type A. If $p,q : x \rightsquigarrow y$ are paths in A and if $s : p \rightsquigarrow q$ is a path in the space $x \rightsquigarrow y$ then there exists, for every $u \in P(x)$, a path $s \cdot u : p \cdot u \rightsquigarrow q \cdot u$ in $P(b)$.*

PROOF. Let $D(p,q,s)$ be the space $\prod(u : P(x)),\ p \cdot u \rightsquigarrow q \cdot u$. Then we have, for each path $p : x \rightsquigarrow y$, that $D(p,p,\mathsf{id}_p) = \prod(u : P(x)),\ p \cdot u \rightsquigarrow p \cdot u$ has the term

$$d(p) :\equiv \lambda u.\mathsf{id}_{p \cdot u}$$

and hence the induction principle for paths gives us a path $s \cdot u : p \cdot u \rightsquigarrow q \cdot u$ for each $s : p \rightsquigarrow q$. ∎

Lemma 2.2.12. *Suppose that $p,q : x \rightsquigarrow y$ are paths in a type A and that $s : p \rightsquigarrow q$ is a path in $x \rightsquigarrow y$. Then we have, for any path $h : w \rightsquigarrow x$ in A, a path $s \bullet\!\!- h : p \bullet h \rightsquigarrow q \bullet h$ and we have for any path $k : y \rightsquigarrow z$ a path $k \bullet\!\!- s : k \bullet p \rightsquigarrow k \bullet q$.*

PROOF. The two assertions are clearly similar, so we only prove the existence of $s \bullet\!\!- h : p \bullet h \rightsquigarrow q \bullet h$. Let D be the dependent type over Id_A given by

$$D(p,q,s) :\equiv \prod\prod(w : A)(h : w \rightsquigarrow x),\ p \bullet h \rightsquigarrow q \bullet h.$$

Then $D(p,p,\mathsf{id}_p)$ has the term $\lambda w.h.\mathsf{id}_{p \bullet h}$, so the induction principle for paths gives us a term of type $D(p,q,s)$ for each path $s : p \rightsquigarrow q$ in the space $x \rightsquigarrow y$. ∎

Lemma 2.2.13. *Suppose that $p,q : x \rightsquigarrow y$ are paths in A. Then for every path $s : p \rightsquigarrow q$ there is a path $s^{[-1]} : p^{-1} \rightsquigarrow q^{-1}$.*

PROOF. Immediate with induction on s. ∎

2.2.5 Paths and dependent sums

The following lemma is of crucial importance. It shows what is needed to construct a path in a total space.

Lemma 2.2.14. *Suppose P is a dependent space over A. To give a path from $\langle x,u\rangle$ to $\langle y,v\rangle$ in the total space $\sum(x:A),\ P(x)$ of P it suffices to give a path $\gamma_0 : x \rightsquigarrow y$ in A and a path $\gamma_1 : \gamma_0 \cdot u \rightsquigarrow v$ in $P(b)$. Thus, we get a function of type*

$$\prod(x,y:A)(u:P(x))(v:P(y))(p:x\rightsquigarrow y),\ (p\cdot u\rightsquigarrow v)\to(\langle x,u\rangle\rightsquigarrow\langle y,v\rangle).$$

PROOF. Suppose that $\gamma_0 : x \rightsquigarrow y$ and that $\gamma_1 : \gamma_0 \cdot u \rightsquigarrow v$. Then $\langle b, \gamma_1\rangle \bullet \gamma_{0\Sigma}$ is a path from $\langle x,u\rangle$ to $\langle y,v\rangle$. Recall that $\gamma_{0\Sigma}$ is obtained with the total path lemma 2.1.8. ∎

Lemma 2.2.15. *Suppose that P is a dependent space over A and that $\gamma : w \rightsquigarrow w'$ is a path in the total space $\sum(x:A),\ P(x)$ of P. Then there are paths*

$$\gamma_0 : \mathsf{proj}_1\, w \rightsquigarrow \mathsf{proj}_1\, w'$$
$$\gamma_1 : \gamma_0 \cdot \mathsf{proj}_2\, w \rightsquigarrow \mathsf{proj}_2\, w'.$$

PROOF. We can take $\gamma_0 :\equiv \mathsf{proj}_1\, \gamma$. The path γ_1 is found with path induction over γ. ∎

Lemma 2.2.16. *Suppose that A is a type, $P : A \to \mathsf{Type}$ is a dependent type over A and that $Q : \prod(x:A),\ P(a) \to \mathsf{Type}$ is a dependent type over P. Then any path $p : x \rightsquigarrow y$ in A induces a dependent function $\prod(u:P(x)),\ Q(x,u) \to Q(y, p\cdot u)$.*

PROOF. This is immediate with induction on p. ∎

The following lemma shows how to compute the transport when the dependent space is itself a dependent sum:

Lemma 2.2.17. *Suppose that $B : A \to \mathsf{Type}$ is a dependent type over A and that $Q : (\sum(x:A),\ B(x)) \to \mathsf{Type}$ is a dependent type over $\sum(x:A),\ B(x)$ and let P be the dependent type over A given by*

$$P(x) :\equiv \sum(v:B(x)),\ Q(x,v).$$

Then for any path $p : x \rightsquigarrow y$ and any $\langle v,q\rangle : P(x)$ there is a path

$$p\cdot\langle v,q\rangle \rightsquigarrow \langle p\cdot v, p_\Sigma(v)\cdot q\rangle,$$

where the transportation in $p\cdot v$ is taken with respect to B and where $p_\Sigma(v) : \langle x,v\rangle \rightsquigarrow \langle y, p\cdot v\rangle$ is the path in $\sum(x:A),\ B(x)$ above p given by the path lifting property.

PROOF. Immediate with induction on p. ∎

The above lemma also has a version where B is non-dependent.

Lemma 2.2.18. *Let A, B, Q and P be as in the previous lemma. When B is constant there is, for any path $p : x \rightsquigarrow y$ and any $\langle v, q\rangle : P(x)$, a path*

$$p \cdot \langle v, q\rangle \rightsquigarrow \langle v, p \cdot q\rangle,$$

where the transport in $p \cdot q$ is taken with respect to the dependent type $\lambda x.Q(x, v)$ over A.

PROOF. The claim follows with induction on p. ∎

2.3 Homotopy type theory

In this section we will introduce ideas from homotopy theory to type theory with identity types. The basic notions of *homotopy*, *contractibility* and *homotopy fiber* will be introduced here. These notions are fundamental in the understanding of equivalences. To help explaining the definition of homotopy we give below, we also take a brief look at *naive* function extensionality. We will not assume any form of function extensionality in this section, unless we are explicit about it.

2.3.1 Homotopies

Lemma 2.3.1. *Suppose that $f, g : \prod(x : A),\ P(x)$ are sections of a dependent space P over A and let $\alpha : f \rightsquigarrow g$. Then α determines paths $\alpha(x) : f(x) \rightsquigarrow g(x)$ for each $x : A$ and the square*

$$\begin{array}{ccc} f(y) & \xrightarrow{\alpha(y)} & g(y) \\ {\scriptstyle f(p)}\uparrow & & \uparrow{\scriptstyle g(p)} \\ p \cdot f(x) & \xleftarrow[(p\cdot\alpha(x))^{-1}]{} & p \cdot g(x) \end{array}$$

commutes.

PROOF. Both parts of the assertion follow with induction on α. ∎

The previous lemma gives a dependent function of type

$$(f \rightsquigarrow g) \to \prod(x : A),\ f(x) \rightsquigarrow g(x)$$

for any $f, g : \prod(x : A),\ P(x)$. Thus, the type $\prod(x : A),\ f(x) \rightsquigarrow g(x)$ is a weaker notion than the type $f \rightsquigarrow g$. The naive function extensionality principle is a function in the other direction.

Definition 2.3.2. The naive function extensionality principle is that there is a function

$$\left(\prod(x : A),\ f(x) \rightsquigarrow g(x)\right) \to (f \rightsquigarrow g)$$

for any pair f, g of sections of a dependent type P over A. ♦

Definition 2.3.3. Two sections f and g of a dependent type P over a type A are said to be homotopic if there is an element H of type

$$\prod(x:A),\ f(x) \rightsquigarrow g(x).$$

In that case, H is said to be a homotopy from f to g. We denote the space of homotopies from f to g by $f \sim g$. ♦

The reader might have expected that a homotopy between to functions f and g would be defined as a path from f to g, and not merely a choice of paths $f(x) \rightsquigarrow g(x)$ for every $x:A$. After all, the latter would not quite suffice in classical homotopy theory. But the reason that this would not suffice in classical homotopy theory is that the choice of paths $f(x) \rightsquigarrow g(x)$ could be made *in a discontinuous way*. This would not be possible in any interpretation of intensional type theory into topology, because we know from the dependent map lemma that functions act on paths as well, which is a form of continuity. This vaguely indicates that our definition of homotopies might indeed be sensible. The real reason that this is a sensible definition is that the models we have in mind (mainly the model of simplicial sets) satisfy the function extensionality principle we have stated above (even in a stronger form, but we will come to that later). With function extensionality, homotopies induce (and come from) paths, which brings us back to our original intuition of what a homotopy should be. We will postpone a detailed discussion on functional extensionality to section 2.5.

Definition 2.3.4. Lemma 2.3.1 gives a function

$$\mathsf{hApply} : \prod \{A : \mathsf{Type}\}\{P : A \to \mathsf{Type}\}\big(f,g : \prod(x:A),\ P(x)\big),\ (f \rightsquigarrow g) \to (f \sim g). \quad ♦$$

There is also a variant of the commutativity of the square in lemma 2.3.1.

Lemma 2.3.5. *Suppose that $H : f \sim g$ is a homotopy for two sections $f,g : \prod(x : A),\ P(x)$ of a dependent type P over A. The square*

$$\begin{array}{ccc} f(y) & \xrightarrow{\ H(y)\ } & g(y) \\ {\scriptstyle f(p)}\uparrow & & \uparrow{\scriptstyle g(p)} \\ p\cdot f(x) & \xleftarrow[(p\cdot H(x))^{-1}]{} & p\cdot f(y) \end{array}$$

commutes for each path $p : x \rightsquigarrow y$ in A.

PROOF. Immediate with induction on p. ∎

Definition 2.3.6. If H is a homotopy from f to g and if K is a homotopy from g to h, then we define the homotopy $K \bullet H$ by

$$K \bullet H(x) := K(x) \bullet H(x).$$

For every function f there is an identity homotopy id_f and any homotopy H has an inverse H^{-1} defined pointwise by $H^{-1}(x) := H(x)^{-1}$. ♦

Note that even though we have not described equivalence relations in type theory yet, it is intuitively clear that being homotopic is an equivalence relation on the type $\prod(x:A),\ P(x)$ of sections of P.

So far we have considered the most general case where f and g are sections of a dependent type P over A. Homotopies between non-dependent functions are just a special case of homotopies between dependent functions. Besides the composition $\bullet$ which we have described above, homotopies of non-dependent functions may also be composed horizontally.

Definition 2.3.7. Suppose that $f,g:A\to B$ and $f',g':B\to C$ are non-dependent functions. If $H:f\sim g$ and $H':f'\sim g'$ are homotopies, we may compose H with H' horizontally by defining

$$H\circ H' :\equiv \lambda a.g'(H(a))\bullet H'(f(a)).$$

Thus, being homotopic is a congruence relation for non-dependent functions. ⧫

Remark 2.3.8. Notice that we could also have defined $H\circ H'$ to be the function

$$\lambda a.H'(g(a))\bullet f'(H(a)).$$

This composition is not propositionally equal to the original horizontal composition, but rather homotopic to it. ⋆

The following lemma is a non-dependent version of lemma 2.3.5.

Lemma 2.3.9. *Suppose that $H:f\sim g$ is a homotopy between functions $f,g:A\to B$. Then the square*

$$\begin{array}{ccc} f(y) & \overset{H(y)}{\rightsquigarrow} & g(y) \\ {\scriptstyle f(p)}\uparrow & & \uparrow{\scriptstyle g(p)} \\ f(x) & \underset{H(x)}{\rightsquigarrow} & g(x) \end{array}$$

commutes for every path $p:x\rightsquigarrow y$ in A.

PROOF. Immediate with induction on p. ■

Definition 2.3.10. A function $f:A\to B$ is said to be a homotopy isomorphism if there exists a function $g:B\to A$ with the property that $\mathsf{id}_A\sim g\circ f$ and $f\circ g\sim \mathsf{id}_B$. In this case g is said to be an inverse for f. The space $\mathsf{iso}(A,B)$ of homotopy isomorphisms from A to B is the space

$$\sum(f:A\to B)(g:B\to A),\ (\mathsf{id}_A\sim g\circ f)\times(f\circ g\sim \mathsf{id}_B).$$ ⧫

Lemma 2.3.11. *Any two inverses of a function $f:A\to B$ are homotopic.*

PROOF. Suppose that $\langle f, g, H, K\rangle$ and $\langle f, g', H', K'\rangle$ are elements of $\mathsf{iso}(A, B)$ and let $b : B$. Then $K(b) : f(g(b)) \rightsquigarrow b$ and $K'(b) : f(g'(b)) \rightsquigarrow b$. Thus we may apply g to the path $K'(b)^{-1} \bullet K(b)$, from which we get a path $g(b) \rightsquigarrow g'(b)$ as the composition

$$g(b) \overset{H(g(b))}{\rightsquigarrow} g(f(g(b))) \overset{g(K'(b)^{-1} \bullet K(b))}{\rightsquigarrow} g(f(g'(b))) \overset{H(g'(b))^{-1}}{\rightsquigarrow} g'(b)$$

This shows that g and g' are homotopic. ∎

Lemma 2.3.12. *A function is an isomorphism whenever it is homotopic to an isomorphism.*

PROOF. This follows directly from the operations defined in definition 2.3.7. ∎

Lemma 2.3.13. *Suppose that $\langle f, g, H, K\rangle$ is a term of type $\mathsf{iso}(A, B)$. Then H and K are related via the squares*

$$\begin{array}{ccc} f(a) & \overset{p}{\rightsquigarrow} & b \\ {\scriptstyle f(H(a))}\downarrow & & \uparrow{\scriptstyle K(b)} \\ f(g(f(a))) & \underset{f(g(p))}{\rightsquigarrow} & f(g(b)) \end{array}$$

which commute for every $a : A$ and $p : f(a) \rightsquigarrow b$.

2.3.2 Contractible spaces

Definition 2.3.14. A space A is said to be contractible if the space

$$\mathsf{isContr}(A) :\equiv \sum(a : A) \prod(x : A),\ x \rightsquigarrow a$$

is inhabited. A pair $\langle a, \lambda x.p(x)\rangle$ is then called a witness of the contractibility of A. If $\langle a, \lambda x.p(x)\rangle$ is a witness of the contractibility of A, then a is said to be the center of contraction. ⧫

Thus, a space A is contractible if there is an element $a : A$ and a homotopy from $\lambda x.x$ to $\lambda x.a$. Note that contractible spaces are always non-empty.

Again, we should remark that in classical homotopy theory, the notion 'there exists an element $a \in A$ such that for every $x \in A$ there is a path from x to a' is *not* a notion of contractibility. For instance, the circle has this property. Rather, the notion of contractibility is that there exists a point $a \in A$ such that the identity map on A is homotopic to the constant map sending every element to a, which is also the correct interpretation of the type theoretical notion of contractibility we have presented above.

Lemma 2.3.15. *The type* unit *is contractible.*

PROOF. We prove this with induction. Let P be the dependent type over unit given by $P(x) \equiv x \rightsquigarrow \mathsf{tt}$. Then $\mathsf{id}_{\mathsf{tt}} : P(\mathsf{tt})$, so the induction principle for unit gives a section $f : \prod(x : \mathsf{unit}),\ P(x)$. Hence the pair $\langle \mathsf{tt}, f \rangle$ is a term of type $\mathsf{isContr}(\mathsf{unit})$. ∎

Lemma 2.3.16. *For any space A and any $a : A$, the space $\sum(x : A),\ x \rightsquigarrow a$ is contractible.*

PROOF. Note that whenever $p : x \rightsquigarrow a$ it holds that that $p \cdot p \rightsquigarrow \mathsf{id}_a$ by lemma 2.2.9, so lemma 2.2.14 gives us the desired result. ∎

Lemma 2.3.17. *If A is a contractible space, then so is Id_A.*

PROOF. If A is contractible there exists an $a : A$ such that $\mathsf{id}_A \sim \lambda x.a$. The claim follows immediately by **??** and **??**. ∎

Lemma 2.3.18. *Suppose that A is a space with the property that $x \rightsquigarrow y$ is contractible for each $x, y : A$. Then we have*

$$A \to \mathsf{isContr}(A).$$

PROOF. Suppose that

$$\varphi : \prod(x, y : A),\ \mathsf{isContr}(x \rightsquigarrow y)$$

and that $a : A$. Then $\lambda x.\, \mathsf{proj}_1\, \varphi(x, a)$ is a term of $\mathsf{isContr}(A)$. ∎

2.3.3 Homotopy fibers

The following defines the 'inverse image' for a function $f : A \to B$.

Definition 2.3.19. Suppose $f : A \to B$ is a function from A to B. The homotopy fiber of a point $b : B$, denoted by $\mathsf{hFiber}(f, b)$, is the space

$$\sum(a : A),\ f(a) \rightsquigarrow b.$$

Note that $\mathsf{hFiber}(f, b)$ is exactly the total space of the dependent type $P(a) \equiv f(a) \rightsquigarrow b$ over A. ♦

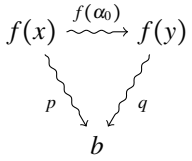
Definition 2.3.20. A function $f : A \to B$ is said to be essentially surjective if there exists a dependent function of type

$$\prod(b : B),\ \mathsf{hFiber}(f, b)$$

and f is said to be essentially injective if there exists a dependent function of type

$$\prod(x, y : A),\ (f(x) \rightsquigarrow f(y)) \to (x \rightsquigarrow y).$$ ♦

Lemma 2.3.21. *Suppose that $f : A \to B$ is a function and that $b : B$. Then for any path $\alpha : \langle x, p\rangle \rightsquigarrow \langle y, q\rangle$ in $\mathsf{hFiber}(f, b)$ there is a commuting triangle*



(1)

of paths in B, where $\alpha_0 : x \rightsquigarrow y$ is the base path of α. Conversely, to give a path $\langle x, p\rangle \rightsquigarrow \langle y, q\rangle$ in $\mathsf{hFiber}(f, b)$ it suffices to find a path $\alpha_0 : x \rightsquigarrow y$ such that the triangle in (1) commutes.

PROOF. Note that $\alpha_0 = \mathsf{id}_{x_0}$ if $\alpha = \mathsf{id}_{\langle x,p\rangle}$, so the the commutativity of the triangle in (1) follows immediately from an application of induction on α.

For the converse we will first show that for any path $\alpha_0 : x \rightsquigarrow y$ in A and any $p : f(x) \rightsquigarrow b$ in B there is a path from $\alpha_0 \cdot p$ to $p \bullet f(\alpha_0)^{-1}$. Let $D(x, y, \alpha_0)$ be the type

$$\prod (b : B)(p : f(x) \rightsquigarrow b),\ \alpha_0 \cdot p \rightsquigarrow p \bullet f(\alpha_0)^{-1},$$

where the transportation is taken with respect to the dependent type $P(x) = f(x) \rightsquigarrow b$. Then $D(x, x, \mathsf{id}_x)$ contains the canonical term $\lambda(b, p) \rightsquigarrow \mathsf{reflRight}(p)$. Thus, by the induction principle for paths it follows that there is a path from $\alpha_0 \cdot p$ to $p \bullet f(\alpha_0)^{-1}$ for every $\alpha_0 : x \rightsquigarrow y$ and $p : f(x) \rightsquigarrow b$.

To find a path from $\langle x, p\rangle$ to $\langle y, q\rangle$ it suffices to find a path from $\alpha_0 : x \rightsquigarrow y$ in A and a path $\alpha_1 : \alpha_0 \cdot p \rightsquigarrow q$ in the space $f(y) \rightsquigarrow b$. Since there is a path from $\alpha_0 \cdot p$ to $p \bullet f(\alpha_0)^{-1}$), it suffices to find a path from $p \bullet f(\alpha_0)^{-1}$ to q, which we get from the commutativity of (1). ∎

Lemma 2.3.22. *Lemma 2.3.21 determines for each $f : A \to B$, $b : B$, $x, y : A$ and $u : f(x) \rightsquigarrow b$ and $v : f(y) \rightsquigarrow v$, a function*

$$\varphi : \Big(\sum (p : x \rightsquigarrow y),\ v \bullet f(p) \rightsquigarrow u\Big) \to \langle x, u\rangle \rightsquigarrow \langle y, v\rangle.$$

To find a path from $\varphi(p, q)$ to $\varphi(p', q')$, where $p, p' : x \rightsquigarrow y$, $q : v \bullet f(p) \rightsquigarrow u$ and $q' : v \bullet f(p') \rightsquigarrow v$, it suffices to find a path $\mu_0 : p \rightsquigarrow p'$ and a path $\mu_1 : q \bullet (v \bullet f(\mu_0)^{-1}) \rightsquigarrow q'$.

PROOF. For this, we only need to show that there is a path from $q \bullet (v \bullet f(\mu_0)^{-1})$ to $\mu_0 \cdot q$. This is immediate with induction on μ_0. ∎

Lemma 2.3.23. *Suppose that $f : A \to B$ is a function, that $q : b \rightsquigarrow b'$ is a path in B and that $\langle a, p\rangle : \mathsf{hFiber}(f, b)$. Then $q \cdot \langle a, p\rangle \rightsquigarrow \langle a, q \bullet p\rangle$. In particular, $p \cdot \langle a, \mathsf{id}_{f(a)}\rangle \rightsquigarrow \langle a, p\rangle$.*

PROOF. Immediate with induction on q. ∎

2.4 Equivalences

With the homotopy fiber, we are able to define equivalences of types.

2.4.1 Definition and first applications of equivalences

Definition 2.4.1. A function $f : A \to B$ is said to be an equivalence if the homotopy fiber $\mathsf{hFiber}(f, b)$ is contractible for every $b : B$. Thus, $f : A \to B$ is an equivalence if there is a term of type

$$\mathsf{isEquiv}(f) := \prod(b : B),\ \mathsf{isContr}(\mathsf{hFiber}(f, b))$$

Two spaces A and B are said to be equivalent if the space $\sum(f : A \to B),\ \mathsf{isEquiv}(f)$ is inhabited; we denote this type by $\mathsf{equiv}(A, B)$ or just by $A \simeq B$. ♦

Lemma 2.4.2. *The identity map* $\mathsf{id}_A : A \to A$ *is always an equivalence.*

PROOF. We have to show that $\mathsf{hFiber}(\mathsf{id}_A, a)$, which is the total space of the dependent type $\lambda x.(x \rightsquigarrow a)$ over A, is contractible for every element $a : A$. If $p : x \rightsquigarrow a$ is a path in A, then the total path $p_\Sigma(p)$ goes from $\langle x, p\rangle$ to $\langle a, p \cdot p\rangle$. By lemma 2.2.9 there is a path from $p \cdot p$ to id_a in the type $a \rightsquigarrow a$. The composition of these yields a path from $\langle x, p\rangle$ to $\langle a, \mathsf{id}_a\rangle$, which shows that $\mathsf{hFiber}(\mathsf{id}_A, a)$ is contractible for each $a : A$. ∎

Corollary 2.4.3. *Suppose P is a dependent type over A and that $p : x \rightsquigarrow y$ is a path in A. Then transportation along p is an equivalence from $P(x)$ to $P(y)$.*

PROOF. Let $D(x, y, p)$ be the type $\mathsf{isEquiv}(\mathsf{transport}(P, p))$. By lemma 2.4.2, the space $D(x, x, \mathsf{id}_x)$ is inhabited since the identity map $\mathsf{id}_{P(x)} : P(x) \to P(x)$ is an equivalence for all $x : A$. Thus, the induction principle for identity types gives us a term of $D(x, y, p)$ for any $p : x \rightsquigarrow y$ in A, which shows that $\mathsf{transport}(P, p)$ is an equivalence for any path p. ∎

Lemma 2.4.4. *A space is contractible whenever it is equivalent to a contractible space.*

PROOF. Suppose that $f : A \to B$ is an equivalence and that

$$E : \prod(b : B),\ \mathsf{isContr}(\mathsf{hFiber}(f, b))$$

and suppose that B is contractible with $\beta : \sum(b : B) \prod(y : B),\ y \rightsquigarrow b$. We have seen that the pair $\langle f, E\rangle$ induces an equivalence $\langle f^{-1}, E^{-1}\rangle$ from B to A. Thus, to show that A is contractible it suffices to find a dependent function of type $\prod(x : A),\ x \rightsquigarrow f^{-1}(\mathsf{proj}_1(\beta))$. Note that the dependent function $\mathsf{proj}_2(\beta) : \prod(y : B),\ y \rightsquigarrow \mathsf{proj}_1(\beta)$ induces the dependent function $f^*(\mathsf{proj}_2(\beta)) : \prod(x : A),\ f(x) \rightsquigarrow \mathsf{proj}_1(\beta)$ given by

$$f^*(\mathsf{proj}_2(\beta))(x) = \mathsf{proj}_2(\beta)(f(x))$$

We may now define $\alpha : \prod(x : A),\ x \rightsquigarrow f^{-1}(\mathsf{proj}_1(\beta))$ by

$$\alpha(x) = f^{-1}(f^*(\mathsf{proj}_2(\beta)(x)) \bullet \eta(x),$$

which shows that A is indeed contractible. ∎

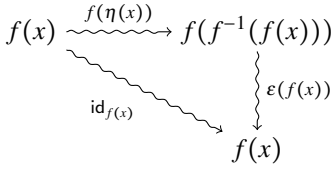
2.4.2 Homotopy isomorphisms are equivalences

It is now time to investigate the anatomy of equivalences somewhat closer. In doing so, we will establish properties of equivalences that will make it much easier for us to find equivalences.

Lemma 2.4.5. *Suppose that $f : A \to B$ is an equivalence. Then there exists a function $f^{-1} : B \to A$ for which there are homotopies*

$$\eta : \prod(x : A),\ x \rightsquigarrow f^{-1}(f(x))$$
$$\varepsilon : \prod(y : B),\ f(f^{-1}(y)) \rightsquigarrow y$$

such that the triangle



(2)

commutes for every $x : A$.

Remark 2.4.6. Our notation of f^{-1} here is sloppy: what we really mean is that if $\langle f, E\rangle : \mathsf{equiv}(A, B)$, then there is a function $\langle f, E\rangle^{-1} : B \to A$. In other words, the proof k that f is an equivalence is essential in the definition of the map f^{-1}. We will prove later that f^{-1} is also an equivalence. ⋆

PROOF. Suppose that $f : A \to B$ is an equivalence and that $\lambda y.E(y)$ is a term of the space $\mathsf{isEquiv}(f)$. Then $E(y)$ is a term of type $\sum(u : \mathsf{hFiber}(f, y))\prod(v : \mathsf{hFiber}), v \rightsquigarrow u$, so $\mathsf{proj}_1\, E(y)$ is a term of type $\mathsf{hFiber}(f, y)$, which yields the term $\mathsf{proj}_1(\mathsf{proj}_1\, E(y))$ of type A. Thus, the function $f^{-1} := \lambda y.\, \mathsf{proj}_1(\mathsf{proj}_1\, E(y))$ is of type $B \to A$. Note that since $\mathsf{proj}_1\, E(y)$ is in $\mathsf{hFiber}(f, y)$, we have the path

$$\varepsilon(y) := \mathsf{proj}_2(\mathsf{proj}_1\, E(y)) : f(g(y)) \rightsquigarrow y.$$

To find η we have to look at the second projection of $E(f(x))$:

$$\mathsf{proj}_2\, E(f(x)) : \prod(u : \mathsf{hFiber}(f, f(x)),\ u \rightsquigarrow \mathsf{proj}_1\, E(f(x)).$$

There is, of course, the element $\langle x, \mathsf{id}_{f(x)}\rangle$ in $\mathsf{hFiber}(f, f(x))$, so we have

$$\mathsf{proj}_2\, E(f(x))(\langle x, \mathsf{id}_{f(x)}\rangle) : \langle x, \mathsf{id}_{f(x)}\rangle \rightsquigarrow \mathsf{proj}_1\, E(f(x)).$$

Note that this is a path in the homotopy fiber of f at $f(x)$, hence we can apply lemma 2.3.21 to conclude that there is a path

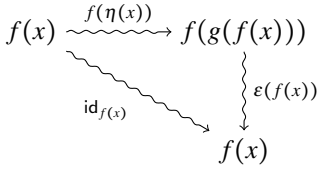
$$\eta(x) : x \rightsquigarrow f^{-1}(f(x)).$$

such that the triangle in 2 commutes. ■

Corollary 2.4.7. *Every equivalence is an isomorphism.*

We gather the data of lemma 2.4.5 in a new definition:

Definition 2.4.8. An adjoint equivalence from A to B is a quintuple $\langle f, g, \eta, \varepsilon, \alpha \rangle$ consisting of a function $f : A \to B$, a function $g : B \to A$, a homotopy $\eta : \mathsf{id}_A \sim g \circ f$, a homotopy $\varepsilon : f \circ g \sim \mathsf{id}_B$ and paths $\alpha(x)$ witnessing the commutativity of the triangle



for every $x : A$. ♦

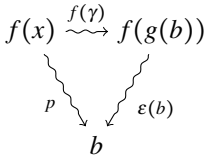
Thus, in lemma 2.4.5 we have shown that every equivalence induces an adjoint equivalence. The converse is also true and it gives us an alternative way to see that a function is an equivalence.

Theorem 2.4.9. *Every adjoint equivalence induces an equivalence.*

PROOF. Suppose that $\langle f, g, \eta, \varepsilon, \alpha \rangle$ is an adjoint equivalence from A to B. We will show that f is an equivalence. Unfolding the definition of equivalence, we have to show that

$$\prod(y : B),\ \mathsf{isContr}(\mathsf{hFiber}(f, b)).$$

Suppose that $y : B$. Our first task is to find a center of contraction in $\mathsf{hFiber}(f, b)$. Note that for $g(b)$ we have a path $\varepsilon(b) : f(g(b)) \rightsquigarrow b$, so the pair $\langle g(b), \varepsilon(b) \rangle$ is a term of $\mathsf{hFiber}(f, b)$. Now suppose that $\langle x, p \rangle$ is another term of $\mathsf{hFiber}(f, b)$. By lemma 2.3.21, we have to find a path $\gamma : x \rightsquigarrow g(b)$ such that the triangle



commutes. As the path γ we take $g(p) \bullet \eta(x)$. Then

$$f(\gamma) = f(g(p) \bullet \eta(x)) \rightsquigarrow f(g(p)) \bullet f(\eta(x)).$$

Note that there is, for every $p : b' \rightsquigarrow b$ in B, a canonical path $p \bullet \varepsilon(b') \rightsquigarrow \varepsilon(b) \bullet f(g(p))$ (to see this, use induction on p). Hence can use α to conclude that there is a path

$$\varepsilon(b) \bullet f(g(p)) \bullet f(\eta(x)) \rightsquigarrow p \bullet \varepsilon(f(x)) \bullet f(\eta(x)) \rightsquigarrow p,$$

which establishes the commutativity of the above triangle. ■

Using function extensionality, we will also be able to show that the space of equivalences form A to B is itself equivalent to the space of adjoint equivalences from A to B. We will postpone the proof of this fact until after we have covered function extensionality in more detail.

We have remarked that the above theorem gives another way to show that a function $f : A \to B$ is an equivalence. That is, we can show that f is an equivalence by finding the appropriate g, η, ε and α. In fact, to show that $f : A \to B$ is an equivalence, we can do without the commutative triangle. This means that to show that f is an equivalence, it suffices to show that f is an isomorphism. However, the space of isomorphisms from A to B is *not* equivalent to the space of equivalences from A to B. Nevertheless, it is a very useful fact which we will exploit over and over again.

Theorem 2.4.10. *Every isomorphism is an equivalence.*

PROOF. Suppose that $f : A \to B$ is an isomorphism, i.e. that $\langle f, g, H, K\rangle$ is an element of the space $\mathsf{iso}(A, B)$. We will find an adjoint equivalence $\langle f, g, \eta, \varepsilon, \alpha\rangle$. First we have to define η, which we take to be

$$\eta(x) :\equiv g(f(H(x)))^{-1} \bullet g(K(f(x)))^{-1} \bullet H(x)$$

and we define $\varepsilon :\equiv K$. Then it is left to verify that $\varepsilon(f(x)) \bullet f(\eta(x)) \rightsquigarrow \mathsf{id}_{f(x)}$, i.e. that the square

$$\begin{array}{ccc} fgfgf(x) & \overset{fgKf(x)}{\rightsquigarrow} & fgf(x) \\ {\scriptstyle fHgf(x)}\uparrow & & \uparrow{\scriptstyle fH(x)} \\ fgf(x) & \underset{Kf(x)}{\rightsquigarrow} & f(x) \end{array}$$

commutes for every $x : A$. We wish to apply lemma 2.3.9 here with the homotopy $Kf : fgf \sim f$ and the path $H(x) : x \rightsquigarrow gf(x)$, but in order to get the above diagram in the right shape we must show that $Hgf(x) \rightsquigarrow gfH(x)$ for each $x : A$ and that $fgK(y) \rightsquigarrow Kfg(y)$ for each $y : A$. Since these are clearly similar, we concentrate only on the first. Note that 2.3.9, applied to the homotopy $H : \mathsf{id}_A \sim gf$ and the path $H(x) : x \rightsquigarrow gf(x)$, gives us the commutativity of the diagram

$$\begin{array}{ccc} gf(x) & \overset{Hgf(x)}{\rightsquigarrow} & gfgf(x) \\ {\scriptstyle H(x)}\uparrow & & \uparrow{\scriptstyle gfH(x)} \\ x & \underset{H(x)}{\rightsquigarrow} & gf(x) \end{array}$$

for each $x : A$. So we have $Hgf(x) \bullet H(x) \rightsquigarrow gfH(x) \bullet H(x)$. Precomposing both with the path $H(x)^{-1}$ reveals that $Hgf(x) \rightsquigarrow gfH(x)$ as desired. ∎

From this theorem, the following corollaries are easy to verify.

Corollary 2.4.11. *The inverse of any equivalence is also an equivalence.*

Corollary 2.4.12. *Equivalences satisfy the 3-for-2 rule: whenever two of $f : A \to B$, $g : B \to C$ and $gf : A \to C$ are equivalences, so is the third.*

Corollary 2.4.13. *A function is an equivalence whenever it is homotopic to an equivalence.*

Corollary 2.4.14. *Any two contractible spaces are equivalent.*

Lemma 2.4.15. *Suppose that P is a dependent type over A. Then we have an equivalence*

$$\mathsf{hFiber}(\mathsf{proj}_1, a) \simeq P(a)$$

for any term a of A.

PROOF. Define the functions

$$\begin{aligned} \varphi &:= \lambda w.(\mathsf{proj}_2\, w) \cdot \mathsf{proj}_2\, \mathsf{proj}_1\, w && : \mathsf{hFiber}(\mathsf{proj}_1, a) \to P(a) \\ \psi &:= \lambda u.\langle\langle a, u\rangle, \mathsf{id}_a\rangle && : P(a) \to \mathsf{hFiber}(\mathsf{proj}_1, a). \end{aligned}$$

We have to verify that $\varphi \circ \psi \sim \mathrm{idmap}$ and that $\psi \circ \varphi \sim \mathrm{idmap}$. For the first, suppose that $u : P(a)$. Then we have that

$$\varphi(\psi(u)) = \varphi(\langle\langle a, u\rangle, \mathsf{id}_a\rangle = \mathsf{id}_a \cdot u = u$$

so we even get that $\varphi \circ \psi = \mathrm{idmap}_{P(a)}$. For the second, we will use the induction principle of dependent sums and show that there is a function

$$\lambda x, u, p.\psi(\varphi(\langle\langle x, u\rangle, p\rangle)) \rightsquigarrow \langle\langle x, u\rangle, p\rangle$$

Note that

$$\psi(\varphi(\langle\langle x, u\rangle, p\rangle)) = \langle\langle a, p \cdot u\rangle, \mathsf{id}_a\rangle.$$

A path from $\langle\langle a, p \cdot u\rangle, \mathsf{id}_a\rangle$ to $\langle\langle x, u\rangle, p\rangle$ is found using path induction over p. ∎

2.4.3 Equivalences of total spaces and fiberwise equivalences

In this final subsection of the section about equivalences we will show that if $\tau : \prod(x : A),\ P(x) \to Q(x)$ is a transformation from P to Q, then τ induces an equivalence between the total spaces whenever $\tau(x)$ is an equivalence for each $x : A$. We will also show the converse of this result, which will play a role in the next section about function extensionality.

Definition 2.4.16. Suppose that P and Q are dependent types over A. We say that a function $\tau : \prod(x : A),\ P(x) \to Q(x)$ is a fiberwise equivalence from P to Q if $\tau(x)$ is an equivalence for each $x : A$. If τ is a fiberwise equivalence from P to Q, we denote the fiberwise equivalence $\lambda x.\tau(x)^{-1}$ from Q to P by τ^{-1} ♦

Definition 2.4.17. Suppose that P and Q are dependent spaces over A and that $\tau : \prod(x : A),\ P(x) \to Q(x)$. Then we define the function

$$\textstyle\sum_A \tau : \sum(x : A),\ P(x) \to \sum(x : A),\ Q(x)$$

by $\lambda w.\langle \mathsf{proj}_1\, w, \tau(\mathsf{proj}_1\, w)(\mathsf{proj}_2\, w)\rangle$. ♦

Lemma 2.4.18. *Suppose that P and Q are dependent spaces over a space A and that there is a fiberwise equivalence τ from P to Q. Then*

$$\textstyle\sum_A \tau : \sum(x : A),\ P(x) \to \sum(x : A),\ Q(x).$$

is an equivalence with inverse $\sum_A \tau^{-1}$.

PROOF. To verify that $\mathsf{id} \sim \sum_A \tau^{-1} \circ \sum_A \tau$, note that

$$\textstyle\sum'_A \tau(\sum_A \tau(\langle x, u\rangle)) = \sum'_A(\langle x, \tau(x, u)\rangle) = \langle x, \tau(x)^{-1}(\tau(x, u))\rangle.$$

A path from $\langle x, u\rangle$ to $\langle x, \tau(x)^{-1}(\tau(x, u))\rangle$ may be given by a path $u \rightsquigarrow \tau(x)^{-1}(\tau(x, u))$ in $P(u)$. For this path we simply take the unit $\eta(x)$ of the adjoint equivalence that $\tau(x)$ induces. The verification that $\sum_A \tau \circ \sum_A \tau^{-1} \sim \mathsf{id}$ follows by symmetry. ■

The following theorem by Voevodsky is a converse to the previous lemma. It is a bit more technical because $(\sum_A \tau)^{-1}$ does not give the candidate inverses $\tau(x)^{-1}$ directly. Yet it is a powerful tool for constructing equivalences, where especially the case in which the total spaces $\sum(x : A),\ P(x)$ and $\sum(x : A),\ Q(x)$ are contractible is useful. Indeed, in that case the only thing we need to conclude that the fibers $P(x)$ are equivalent to $Q(x)$ for each $x : A$ is a transformation from P to Q, i.e. a function $\tau : \prod(x : A),\ P(x) \to Q(x)$. We will use this idea in the proof that the weak function extensionality principle implies the strong function extensionality principle (theorem 2.5.8), in the section on the univalence axiom we will use it to show that functions are total single-valued relations (theorem **??**) and in the next chapter we will use this idea to show that, assuming univalence, the fundamental group of the circle is $\mathbb{Z}$ (theorem **??**).

Theorem 2.4.19. *Suppose that P and Q are dependent spaces over a space A and that $\tau : \prod(x : A),\ P(x) \to Q(x)$ has the property that $\sum_A \tau$ is an equivalence. Then τ is a fiberwise equivalence.*

PROOF. First, we have to define the candidate inverse $\sigma(x) : Q(x) \to P(x)$ of $\tau(x)$ for each $x : A$. Suppose that $x : A$ and $v : Q(x)$. Since $\sum_A \tau$ is an equivalence, it has an inverse $(\sum_A \tau)^{-1}$ which we may apply to the pair $\langle x, v\rangle$ to obtain

$$\langle x', u\rangle := (\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle).$$

Now notice that $\sum_A \tau(\langle x', u\rangle) = \langle x', \tau(x', u)\rangle$ and that the counit ε of the adjoint equivalence given by $\sum_A \tau$ yields the path $\varepsilon(\langle x, v\rangle) : \langle x', \tau(x', u)\rangle \rightsquigarrow \langle x, v\rangle$. Thus, we find $\varepsilon(\langle x, v\rangle)_0 : x' \rightsquigarrow x$, which gives us

$$\sigma(x, v) := \varepsilon(\langle x, v\rangle)_0 \cdot u : P(x).$$

Now we wish to show that $\mathsf{id}_{P(x)} \sim \sigma(x) \circ \tau(x)$ and that $\tau(x) \circ \sigma(x) \sim \mathsf{id}_{Q(x)}$. For $u : P(x)$, notice that a path

$$\varepsilon(\langle x, \tau(x,u)\rangle)_0 \cdot \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, \tau(x,u)\rangle)) \rightsquigarrow u$$

may be given by a path

$$\langle x, \varepsilon(\langle x, \tau(x,u)\rangle)_0 \cdot \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, \tau(x,u)\rangle))\rangle \rightsquigarrow \langle x, u\rangle.$$

This helps, since we now may use the general fact that $\langle x, p \cdot a\rangle \rightsquigarrow \langle y, a\rangle$ whenever $p : y \rightsquigarrow x$ is a path in A and $a : P(y)$, which gives us

$$\begin{aligned}
&\langle x, \varepsilon(\langle x, \tau(x,u)\rangle)_0 \cdot \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, \tau(x,u)\rangle))\rangle \\
&\qquad \rightsquigarrow \langle y, \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, \tau(x,u)\rangle))\rangle \\
&\qquad = (\textstyle\sum_A \tau)^{-1}(\langle x, \tau(x,u)\rangle) \\
&\qquad \rightsquigarrow \langle x, u\rangle,
\end{aligned}$$

where the latter path is given by the counit of the adjunction $\sum_A \tau$. To get a homotopy $\tau(x) \circ \sigma(x) \sim \mathsf{id}_{Q(x)}$, notice that for $v : Q(x)$ we can find a path $\tau(x, \sigma(x,v)) \rightsquigarrow v$ by finding a path

$$\langle x, \tau(x, \varepsilon(\langle x, v\rangle)_0 \cdot \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle))\rangle \rightsquigarrow \langle x, v\rangle.$$

By a similar calculation, we obtain

$$\begin{aligned}
&\langle x, \tau(x, \varepsilon(\langle x, v\rangle)_0 \cdot \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle))\rangle \\
&\qquad \rightsquigarrow \langle y, \varepsilon(\langle x, v\rangle)_0 \cdot \tau(y, \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle))\rangle \\
&\qquad \rightsquigarrow \langle x, \tau(x, \mathsf{proj}_2((\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle))\rangle \\
&\qquad = \textstyle\sum_A \tau((\textstyle\sum_A \tau)^{-1}(\langle x, v\rangle)) \\
&\qquad \rightsquigarrow \langle x, v\rangle
\end{aligned}$$

We have used that $\tau(x, p \cdot a) \rightsquigarrow p \cdot \tau(y, a)$ for each path $p : y \rightsquigarrow x$ and each $a : P(y)$. This fact follows from path induction on p. ∎

Corollary 2.4.20. *Suppose that P and Q are dependent types over A with contractible total spaces, i.e. there are terms*

$$K : \mathsf{isContr}\Big(\textstyle\sum(x:A),\ P(x)\Big) \qquad \textit{and} \qquad L : \mathsf{isContr}\Big(\textstyle\sum(x:A),\ Q(x)\Big).$$

Then there is a term of type $(\prod(x:A),\ P(x) \to Q(x)) \to (\prod(x:A),\ P(x) \simeq Q(x))$.

2.5 The axiom of choice and function extensionality

Suppose we set ourselves the task of proving that the product type $\prod(x:A),\ P(x)$ is contractible, whenever all of the types $P(x)$ are contractible. Then we take two sections

f and g of P and we have to give a path from f to g. This raises a problem, because the hypothesis on P only allows us to construct a homotopy from f to g. Homotopies can be seen as a weaker version of paths in a space of sections; what we really want in this situation is that they induce paths, i.e. that there is a function

$$\prod(A : \mathsf{Type})(P : A \to \mathsf{Type})(f,g : \prod(x:A),\ P(x)),\ (f \sim g) \to (f \rightsquigarrow g)$$

In this section we will show

Definition 2.5.1 (The weak functon extensionality principle). The weak function extensionality principle is that the space

$$\prod(x:A),\ P(x)$$

is contractible for every type A and every dependent type $P : A \to \mathsf{Type}$ such that $P(x)$ is contractible for each $x : A$. In other words, the weak function extensionality principle is a term $\mathsf{wk{-}funExt}$ of type

$$\prod\{A : \mathsf{Type}\}\{P : A \to \mathsf{Type}\}, \Big(\prod(x:A), \mathsf{isContr}(P(x))\Big) \to \mathsf{isContr}\Big(\prod(x:A), P(x)\Big).$$

♦

Our goal in this section is to show that the weak function extensionality principle implies the (strong) function extensionality principle; i.e. that there is a term of type

$$\prod\big\{A : \mathsf{Type}\big\}\big\{P : A \to \mathsf{Type}\big\}\big(f,g : \prod(x:A),\ P(x)\big),\ \mathsf{isEquiv}(\mathsf{hApply}(f,g))$$

This will be established in theorem 2.5.8. *A note:* the Coq repositories on homotopy type theory require also the η-rule in order to prove the strong function extensionality principle from the weak. However, we have assumed the η-rule to hold (even definitionally), so we allow ourselves to freely make use of it. The reason that the Coq repositories mentions it explicitly is that the η-rule is not built in in Coq. In appendix A we justify our approach to the η-rule by presenting product spaces as inductive spaces with a basic constructor λ and with the β-rule as the computation rule.

2.5.1 A weak version of the axiom of choice

Before we begin with the actual investigation of the axiom of choice, we introduce some notation.

Definition 2.5.2. Suppose that P is a dependent type over a type A and that $R : \prod(x : A),\ P(x) \to \mathsf{Type}$ is a dependent type over P. Then we define the types

$$\begin{aligned}\mathsf{section}(R) &:= \prod(x:A)\sum(u : P(x)),\ R(x,u)\\ \mathsf{choice}(R) &:= \sum\big(s : \prod(x:A),\ P(x)\big)\prod(x:A),\ P(x,s(x)).\end{aligned}$$

♦

Lemma 2.5.3 (The axiom of choice). *For every dependent type P over a type A and every dependent type $R : \prod(x:A),\ P(x) \to \mathsf{Type}$ over P, there is a function*

$$\mathsf{ac} : \mathsf{section}(R) \to \mathsf{choice}(R).$$

PROOF. Suppose that $f : \mathsf{section}(R)$. Then

$$s := \lambda x.\mathsf{proj}_1\, f(x)$$

is a section of P with the property that $\prod(x:A),\ R(x,s(x))$. Indeed, the function

$$\lambda x.\mathsf{proj}_2\, f(x)$$

is a term of type $\prod(x:A),\ R(x,s(x))$. ∎

The main goal of this section is to find a condition on R under which ac is an equivalence. Finding a candidate homotopy inverse ac−inv for ac is easy and establishing that ac−inv is its right inverse as well. The difficulty is to show that ac−inv is also a left inverse for ac.

Lemma 2.5.4. *For every dependent type P over a type A and every dependent type $R : \prod(x:A),\ P(x) \to \mathsf{Type}$ over P, there is a function*

$$\mathsf{ac{-}inv} : \mathsf{choice}(R) \to \mathsf{section}(R).$$

PROOF. Suppose that $\langle s, \varphi\rangle$ is a term of type $\mathsf{choice}(R)$. Then

$$\lambda x.\langle s(x), \varphi(x)\rangle$$

is a term of type $\mathsf{section}(R)$. ∎

Lemma 2.5.5. *The function* ac−inv *is a right inverse of* ac.

PROOF. We have to show that $\mathsf{ac} \circ \mathsf{ac{-}inv} \sim \mathsf{id}$. Suppose that $\langle s, \varphi\rangle : \mathsf{choice}(R)$. Then we have

$$\begin{aligned}\mathsf{ac}(\mathsf{ac{-}inv}(\langle s, \varphi\rangle)) &= \mathsf{ac}(\lambda x.\langle s(x), \varphi(x)\rangle)\\ &= \langle \lambda x.s(x), \lambda x.\varphi(x)\rangle.\end{aligned}$$

Hence, from the η-rule it follows that $\mathsf{proj}_1(\mathsf{ac}(\mathsf{ac{-}inv}(\langle s, \varphi\rangle))) = \lambda x.s(x) = s$ and that $\mathsf{proj}_1(\mathsf{ac}(\mathsf{ac{-}inv}(\langle s, \varphi\rangle))) = \lambda x.\varphi(x) = \varphi$. This shows that ac−inv is a right inverse for ac. ∎

The question whether ac−inv is also a left inverse for ac is more subtle. Note that if $f : \mathsf{section}(R)$, then

$$\begin{aligned}\mathsf{ac{-}inv}(\mathsf{ac}(f)) &= \mathsf{ac{-}inv}(\langle \lambda x.\,\mathsf{proj}_1\, f(x), \lambda x.\,\mathsf{proj}_2\, f(x)\rangle)\\ &= \lambda x.\langle \mathsf{proj}_1\, f(x), \mathsf{proj}_2\, f(x)\rangle.\end{aligned}$$

We would like to conclude here that this equals $\lambda x.f(x) = f$, but we have not the means to do so. However, if the spaces

$$\sum(u : P(x)),\ R(x,u)$$

are contractible for each $x : A$, we may apply the weak function extensionality principle, which says in this case that $\mathsf{section}(R)$ is contractible, to derive that there is a path $\lambda x.\langle \mathsf{proj}_1 f(x), \mathsf{proj}_2 f(x)\rangle \rightsquigarrow f$. We slightly generalize this observation in the following lemma.

Lemma 2.5.6. *Suppose that P is a dependent type over a type A and that $R : \prod(x : A),\ P(x) \to \mathsf{Type}$ is a dependent type over P with the property that for every $x : A$ and every*

$$a, b : \sum(u : P(x)),\ R(x,u),$$

the path space $a \rightsquigarrow b$ is contractible. Assuming the weak function extensionality principle, it follows that ac *is an equivalence.*

PROOF. We only have to show that $\mathsf{ac}\text{–}\mathsf{inv} \circ \mathsf{ac} \sim \mathsf{id}$. Suppose that $f : \mathsf{section}(R)$. By the weak function extensionality principle, it suffices to show that

$$\sum(u : P(x)),\ R(x,u)$$

is contractible for each $x : A$. Note that $\sum(u : P(x)),\ R(x,u)$ satisfies the hypothesis of lemma 2.3.18, hence we have a function

$$\Big(\sum(u : P(x)),\ R(x,u)\Big) \to \mathsf{isContr}\Big(\sum(u : P(x)),\ R(x,u)\Big).$$

Since we have $f(x) : \sum(u : P(x)),\ R(x,u)$, it follows that $\sum(u : P(x)),\ R(x,u)$ is contractible. ∎

2.5.2 The strong function extensionality principle from the weak

Recall from lemma 2.3.16, that for any space A and any $a : A$, the space $\sum(x : A),\ x \rightsquigarrow a$ is contractible. If we were to show that homotopies correspond to paths, we would expect similar behavior for homotopies.

Lemma 2.5.7. *Suppose that P is a dependent space over A. Then the space*

$$\sum\big(g : \prod(x : A),\ P(x)\big),\ g \sim f$$

is contractible for each section f of P.

PROOF. Suppose that $f : \prod(x : A),\ P(x)$ is a section of P and let R be the dependent type over P given by $R(x,u) :\equiv u \rightsquigarrow f(x)$. By lemma 2.3.16 it follows that $\sum(u : P(x)),\ u \rightsquigarrow f(x)$ is contractible for each $x : A$. Hence we may apply lemma 2.5.6 to conclude that

$$\Big(\sum\big(g : \prod(x : A),\ P(x)\big),\ g \sim f\Big) \simeq \prod(x : A) \sum(u : P(x)),\ u \rightsquigarrow f(x).$$

On the right hand side, we have a product of contractible spaces, which is contractible by the weak function extensionality principle. ∎

Note that the above lemma nearly gives us the strong function extensionality principle. Reading back all the functions involved in the proof, we see that the center of contraction of

$$\sum\big(g : \prod(x : A),\ P(x)\big),\ g \sim f$$

is the pair $\langle f, \lambda x.\mathsf{id}_{f(x)}\rangle$.

Theorem 2.5.8. *Assuming the weak function extensionality principle, we get a term* funExt *of type*

$$\prod\big\{A : \mathsf{Type}\big\}\big\{P : A \to \mathsf{Type}\big\}\big(f, g : \prod(x : A),\ P(x)\big),\ \mathsf{isEquiv}(\mathsf{hApply}(f, g))$$

PROOF. Suppose that $f : \prod(x : A),\ P(x)$. Recall from theorem 2.4.19 that in order to show that $\prod(g : \prod(x : A),\ P(x)),\ \mathsf{isEquiv}(\mathsf{hApply}(f, g))$, it suffices to show that the function $\lambda g.\mathsf{hApply}(f, g)$ induces an equivalence

$$\big(\sum\big(g : \prod(x : A),\ P(x)\big),\ f \rightsquigarrow g\big) \simeq \sum\big(g : \prod(x : A),\ P(x)\big),\ f \sim g.$$

This follows, since both of these spaces are contractible. ∎

Using the strong function extensionality principle we are able to prove that ac is an equivalence.

Theorem 2.5.9. *For each dependent space $P : A \to \mathsf{Type}$ over a space A and each $R : \prod(x : A),\ (P(x) \to \mathsf{Type})$ over P, we have that* ac *is an equivalence.*

PROOF. Recall that the missing step in the proof that ac is an equivalence was that

$$\lambda x.\langle \mathsf{proj}_1 f(x), \mathsf{proj}_2 f(x)\rangle \rightsquigarrow \lambda x.f(x).$$

Since $\langle \mathsf{proj}_1 f(x), \mathsf{proj}_2 f(x)\rangle \rightsquigarrow f(x)$ for each $x : A$, this follows from the strong function extensionality principle. ∎

2.6 Basic examples of equivalences

In this section we will prove some basic equivalences that will be useful later on. In all of these equivalences, the function extensionality principle is used. Apart from the basic equivalences, we will also start with our program to find correspondence theorems for every space that has an inductive definition. These correspondence theorems assert that there is, for an inductively defined type A and a dependent type P over A, an equivalence between the space of sections of P and the data required by the induction principle to construct a section. They come in a dependent and a non-dependent flavor, and we may use these observations to establish that an inductively defined type is determined uniquely up to equivalence.

In this section, we will find correspondence theorems for dependent sums, the unit and empty types and for identity types. In chapter 3 we will continue to find correspondence theorems for the higher inductive types we define there. With this program in

mind, we will slowly work towards the point of view that the correspondence theorems are a mere reformulation of the induction principles.

We begin with the observation that if P and Q are fiberwise equivalent dependent types over A, then the spaces of sections are also equivalent. This result is analogous to that of lemma 2.4.18 about the equivalence of the total spaces when a fiberwise equivalence is given. However, there is no analogue of theorem 2.4.19 for product spaces.

Lemma 2.6.1. *For two dependent spaces P and Q over A such that $P(x)$ is equivalent to $Q(x)$ for each $x : A$. Then $\prod(x:A),\ P(x)$ and $\prod(x:A),\ Q(x)$ are equivalent.*

PROOF. Define the function $\varphi : \prod(x:A),\ P(x) \to \prod(x:A),\ Q(x)$ by

$$\lambda f.\lambda x.\tau(x, f(x))$$

and define $\psi : \prod(x:A),\ Q(x) \to \prod(x:A),\ P(x)$ by

$$\lambda g.\lambda x.\tau(x)^{-1}(g(x)).$$

To show that $\psi(\varphi(f)) \rightsquigarrow f$ for each $f : \prod(x:A),\ P(x)$ and that $\varphi(\psi(g)) \rightsquigarrow g$ for each $g : \prod(x:A),\ Q(x)$, we will use the function extensionality principle. In the first case, we have to verify that $\psi(\varphi(f))(x) \rightsquigarrow f(x)$ for each $x : A$. By definition, $\psi(\varphi(f))(x) = \tau(x)^{-1}(\tau(x)(f(x)))$; the fact that $\tau(x)$ is an equivalence now gives the desired result. A homotopy $\varphi(\psi(g)) \sim g$ is found similarly. ∎

The following basic equivalence is a correspondence theorem for dependent sums. The equivalence expresses that a section of a dependent type over a sum is uniquely determined by its action on the canonical pairs. In proving this lemma, one sees explicitly the power of the *dependency* in the induction principle.

Lemma 2.6.2 (Correspondence theorem for sums). *Suppose that P is a dependent space over a space A and that $Q : \left(\sum(x:A), P(x)\right) \to \mathsf{Type}$ is a dependent space over $\sum(x:A),\ P(x)$. Then there is an equivalence*

$$\left(\prod(x:A)(u:P(x)),\ Q(\langle x,u\rangle)\right) \simeq \prod\left(w:\sum(x:A),\ P(x)\right),\ Q(w) \tag{3}$$

PROOF. Note that if $f : \prod(x:A)(u:P(x)),\ Q(\langle x,u\rangle)$, we can use the induction principle for $\sum(x:A),\ P(x)$ to find a section $\varphi(f) : \prod(w:\sum(x:A), P(x)),\ Q(w)$ of Q with the property that $\varphi(f)(\langle x,u\rangle) = f(x,u)$ for each $x : A$ and $u : P(x)$. This defines the map

$$\varphi : \prod(x:A)(u:P(x)),\ Q(\langle x,u\rangle) \to \prod\left(w:\sum(x:A),\ P(x)\right),\ Q(w)$$

For $g : \prod\left(w:\sum(x:A),\ P(x)\right),\ Q(w)$ we have

$$\psi(g) :\equiv g \circ \langle -,-\rangle : \prod(x:A)(u:P(x)),\ Q(\langle x,u\rangle).$$

Note that the computation rule of the induction principle for $\sum(x:A),\ P(x)$ automatically gives that $\psi(\varphi(f)) = f$ for each $f : \prod(x:A)(u:P(x)),\ Q(\langle x,u\rangle)$. Therefore, it is

left to show that $\varphi(\psi(g)) \rightsquigarrow g$ for each section g of Q. Using the function extensionality principle, it suffices to show that

$$\prod\big(w : \sum(x:A),\ P(x)\big),\ \varphi(g \circ \langle -,- \rangle) \rightsquigarrow g.$$

In other words, we have to provide a section of a certain dependent type over $\sum(x : A),\ P(x)$. Hence we will use the induction principle. Note that for $x : A$ and $u : P(x)$, we have

$$\varphi(g \circ \langle -,- \rangle)(x,u) = g(\langle x,u \rangle).$$

Thus, the induction principle gives us the desired section. ∎

The following corollary is a non-dependent version of the above statement. It shows how the induction principle for dependent sums is 'like' introducing disjunction on the left.

Corollary 2.6.3. *Suppose that P is a dependent space over A and that B is a space. Then there is an equivalence*

$$\big(\prod(x:A),\ P(x) \to B\big) \simeq \big(\big(\sum(x:A),\ P(x)\big) \to B\big). \tag{4}$$

We use the above equivalences to establish that dependent sums are determined uniquely up to equivalence by their induction principles. Notice how we can think of the space on the right hand side of equation (4) as the space of cocones with vertex B over the collection P of types. The equivalence is the type theoretical analogue of the statement that $\sum(x : A),\ P(x)$ is the vertex of a colimiting cocone. The corollary below states that such vertices are all equivalent and its proof is a direct analogue to the category theoretical proof that any two vertices of colimiting cocones for a diagram are isomorphic. This approach turns out to be fruitful for not only the ordinary inductive spaces that we cover in this section, but also for the higher inductive spaces of the next chapter.

Corollary 2.6.4. *Any two spaces X and Y satisfying the induction principle for $\sum(x : A),\ P(x)$ are equivalent.*

PROOF. Suppose that for a type X there is a function

$$\langle -,- \rangle_X : \prod(x:A),\ P(x) \to X$$

and, as an induction principle, that for each dependent type $\Lambda : X \to \mathsf{Type}$ and each

$$f : \prod(x:A)(u:P(x)),\ \Lambda(\langle x,u \rangle_X)$$

there is a section $s : \prod(w:X),\ \Lambda(x)$ with the property that $s(\langle x,u \rangle_X) = f(x,u)$ for each $x : A$ and $u : P(x)$. Then we have an equivalence

$$\alpha_B : \big(\prod(x:A),\ P(x) \to B\big) \simeq (X \to B)$$

for each type B. Note that $\alpha_X(\langle -,-\rangle_X) \sim \mathsf{id}_X$; one can prove this using the induction principle for X. Making similar assumptions for a space Y, we get a similar equivalence

$$\beta_B : \left(\prod(x:A),\ P(x) \to B\right) \simeq (Y \to B)$$

for each type B. Again, we remark that $\beta_Y(\langle -,-\rangle_Y) \sim \mathsf{id}_Y$. Now we find functions $f : X \to Y$ and $g : Y \to X$ defined by

$$\begin{aligned} f &:= \alpha_Y(\langle -,-\rangle_Y) \\ g &:= \beta_X(\langle -,-\rangle_X). \end{aligned}$$

We wish to show that $f \circ g \rightsquigarrow \mathsf{id}_Y$ and that $g \circ f \rightsquigarrow \mathsf{id}_X$. Since we know that the homotopy fibers of α_B are contractible for each type B and since the homotopy fiber of β_Y^{-1} at $\langle -,-\rangle_Y$ contains id_Y, we note that in order to show that $f \circ g \rightsquigarrow \mathsf{id}_Y$ it suffices to find a path $\beta_Y^{-1}(f \circ g) \rightsquigarrow \langle -,-\rangle_Y$. We use the induction principle for Y, and thus it suffices to verify that $\beta_Y^{-1}(f \circ g)(x,u) \rightsquigarrow \langle x,u\rangle_Y$ for each $x : A$ and $u : P(x)$. This is a matter of calculating, taking the definitions of α and β into account:

$$\begin{aligned} \beta_Y^{-1}(f \circ g)(x,u) &= f \circ g(\langle x,u\rangle_Y) \\ &= \alpha_Y(\langle -,-\rangle_Y)(\beta_X(\rangle -,-\rangle_X)(\langle x,u\rangle_Y)) \\ &= \alpha_Y(\langle -,-\rangle_Y)(\langle x,u\rangle_X) \\ &= \langle x,u\rangle_Y. \end{aligned}$$

This finishes the proof that $f \circ g \rightsquigarrow \mathsf{id}_Y$; the assertion $g \circ f \rightsquigarrow \mathsf{id}_X$ is dealt with by symmetry of the situation. ∎

The correspondence theorem for dependent sums was the most difficult of the correspondence theorems we cover in this section. This is mainly due to the structure of its basic constructors. The cases of unit and empty are, as one might expect, considerably easier. Notice that in the case of unit, we don't have to prove that it is determined uniquely up to equivalence, since we have already established this fact by observing that unit is contractible.

Lemma 2.6.5 (Correspondence theorem for unit**).** *For any dependent type P over the unit type, there is an equivalence*

$$P(\mathsf{tt}) \simeq \prod(x:\mathsf{unit}),\ P(x).$$

PROOF. Suppose $u : P(\mathsf{tt})$; then the induction principle for unit gives a section $\varphi(u) : \prod(x:\mathsf{unit}),\ P(x)$ with the property that $\varphi(u)(\mathsf{tt}) = u$. If $f : \prod(x:\mathsf{unit}),\ P(x)$ is a section of P, define $\psi(f) := f(\mathsf{tt})$. The computation rule for the induction principle immediately gives that $\psi(\varphi(u)) = u$, so it is left to show that $\varphi(\psi(f)) \rightsquigarrow f$ for each section f of P. Using the function extensionality principle, it suffices to show that

$$\prod(x:\mathsf{unit}),\ \varphi(f(\mathsf{tt}))(x) \rightsquigarrow f(x).$$

To show that such a section exists, we use the induction principle for unit once more: note that $\varphi(f(\mathsf{tt}))(\mathsf{tt}) = f(\mathsf{tt})$. ∎

Corollary 2.6.6. *For every space A there is an equivalence*

$$A \simeq (\mathsf{unit} \to A).$$

Lemma 2.6.7 (Correspondence theorem for empty**).** *Suppose P is a dependent type over the empty type* empty*. Then we have an equivalence*

$$\mathsf{unit} \simeq \prod(x : \mathsf{empty}),\ P(x).$$

PROOF. Note that it suffices to show that $\prod(x : \mathsf{empty}),\ P(x)$ is contractible. By the induction principle for the empty type, there is a section f of P. If g is a section of P, we use the function extensionality principle to show that $g \rightsquigarrow f$. A homotopy from g to f is found with the induction principle for empty. ∎

Corollary 2.6.8. *For any type A we have* $\mathsf{unit} \simeq (\mathsf{empty} \to A)$.

Corollary 2.6.9. *Suppose that X has the induction principle that for any dependent type P over X there is a section of P. Then* $X \simeq \mathsf{empty}$.

PROOF. By the above corollary, we get functions $f : \mathsf{empty} \to X$ and $g : X \to$. There is a homotopy $f \circ g \sim \mathsf{id}_X$ by the induction principle of X and there is a homotopy $g \circ f \sim \mathsf{id}_{\mathsf{empty}}$ by the induction principle of empty. ∎

Lemma 2.6.10 (Correspondence theorem for Id_A**).** *Suppose we have a dependent space* $D : \prod(x, y : A),\ (x \rightsquigarrow y) \to \mathsf{Type}$ *over* Id_A *for some type A. Then we have an equivalence*

$$\Big(\prod(x : A),\ D(x, x, \mathsf{id}_x)\Big) \simeq \prod(x, y : A)(p : x \rightsquigarrow y),\ D(x, y, p).$$

PROOF. A function

$$\varphi : \Big(\prod(x : A),\ D(x, x, \mathsf{id}_x)\Big) \to \prod(x, y : A)(p : x \rightsquigarrow y),\ D(x, y, p)$$

is given by the induction principle for Id_A and a function

$$\psi : \Big(\prod(x, y : A)(p : x \rightsquigarrow y),\ D(x, y, p)\Big) \to \prod(x : A),\ D(x, x, \mathsf{id}_x)$$

is given by $\lambda f.\lambda x.f(x, x, \mathsf{id}_x)$. If $d : \prod(x : A),\ D(x, x, \mathsf{id}_x)$, then $\psi(\varphi(d))(x) = d(x)$ by the conversion rule. Hence we get a homotopy $\psi \circ \varphi \sim \mathsf{id}$ from function extensionality. If $f : \prod(x, y : A)(p : x \rightsquigarrow y),\ D(x, y, p)$, we can prove with path induction that there is a homotopy $\varphi(\psi(f)) \sim f$. Indeed, for $x : A$ we have

$$\varphi(\psi(f))(x, x, \mathsf{id}_x) = \psi(f)(x) = f(x, x, \mathsf{id}_x).$$

Therefore, the function extensionality principle gives that $\varphi(\psi(f)) \rightsquigarrow f$ and hence a homotopy $\varphi \circ \psi \sim \mathsf{id}$. ∎

Definition 2.6.11. Suppose that A is a type. A (binary) relation over A is a dependent type $R : A \to A \to \mathsf{Type}$. We say that R is reflexive if there is a function

$$\rho : \prod(x : A),\ R(x,x). \qquad \blacklozenge$$

Corollary 2.6.12. *Suppose that $R : A \to A \to \mathsf{Type}$ is a binary relation over A. Then we have an equivalence*

$$\Big(\prod(x : A),\ R(x,x)\Big) \simeq \prod(x,y : A),\ (x \rightsquigarrow y) \to R(x,y).$$

The previous corollary says, loosely speaking, that for a binary relation R over A, proofs of reflexivity of R are the same as proofs of the assertion that Id_A is contained in R.

Lemma 2.6.13. *Suppose that $R : A \to A \to \mathsf{Type}$ is a relation over A with reflexivity term*

$$\rho : \prod(x : A),\ R(x,x)$$

and with the property that for every dependent type $D : \prod(x,y : A),\ R(x,y) \to \mathsf{Type}$ over R and every $d : \prod(x : A),\ D(x,x,\rho(x))$ there is a section

$$J_R(D,d) : \prod x,y : A(p : R(x,y)),\ D(x,y,p)$$

of R with the property that $J_R(D,d,\mathsf{id}_x) = d(x)$ for each $x : A$. Then we have that

$$(x \rightsquigarrow y) \simeq R(x,y)$$

for each $x,y : A$.

PROOF. Since R satisfies the same induction principle as the identity type on A, it follows that there are equivalences

$$\Big(\prod(x : A),\ S(x,x)\Big) \simeq \prod(x,y : A),\ R(x,y) \to S(x,y)$$

for any $S : A \to A \to \mathsf{Type}$. Therefore, there is a function $f : \prod(x,y : A),\ R(x,y) \to (x \rightsquigarrow y)$ such that $f(x,x,\rho(x)) \rightsquigarrow \mathsf{id}_x$ for each $x : A$. Also, there is a function $g : \prod(x,y : A),\ (x \rightsquigarrow y) \to R(x,y)$ such that $g(x,x,\mathsf{id}_x) \rightsquigarrow \rho(x)$ for each $x : A$.

To show that $g \circ f \sim \mathrm{idmap}$, it suffices to show that $g \circ f$ is in the same homotopy fiber as idmap taken with respect to the above equivalence, i.e. we only have to verify that $g \circ f(x,x,\rho(x)) \rightsquigarrow \rho(x)$. But this is immediate from the properties of f and g we stated. The fact that $f \circ g \sim \mathrm{idmap}$ follows by a similar remark. ∎

We finish this section by stating some more useful equivalences without proof. As most difficult basic equivalences have been constructed, we leave the following to the reader.

Lemma 2.6.14. *For every triple A, B, C of spaces there is an equivalence*

$$(A \to B \to C) \simeq (A \times B \to C).$$

Lemma 2.6.15. *For every triple A, B, C of spaces there is an equivalence*

$$(A \to C) \times (B \to C) \simeq (A + B \to C).$$

Lemma 2.6.16. *Suppose that B is a contractible space. Then for any space A there are equivalences*

$$A \times B \simeq A \quad \textit{and} \quad (B \to A) \simeq A.$$

Lemma 2.6.17. *For any two types A and B, we have $A \times B \simeq B \times A$ and $A + B \simeq B + A$.*

2.7 The univalence axiom

We have seen that the identity map for a type A is always an equivalence. This generalizes to all paths between types:

Lemma 2.7.1. *For every pair A, B of types and every path $p : A \rightsquigarrow B$ in* Type *there is an equivalence $\upsilon(A,B,p) : A \simeq B$.*

PROOF. Let $D(A,B,p)$ be the space $A \simeq B$ of equivalences from A to B. Then idmap_A is a term of $D(A,A,\mathsf{id}_A)$, so by the induction principle for identity types there is a dependent function of type

$$\prod(A,B : \mathsf{Type}),\ (A \rightsquigarrow B) \to (A \simeq B). \qquad \blacksquare$$

Definition 2.7.2. The univalence axiom is the statement that $\upsilon(A,B)$ is an equivalence for all types A and B. In other words, assuming the univalence axiom is assuming that there is a term

$$\mathsf{univalence} : \prod(A,B : \mathsf{Type}),\ \mathsf{isEquiv}(\upsilon(A,B)). \qquad \blacklozenge$$

Using the univalence axiom, we can show that there is an induction principle for equivalences similar to that for paths. This insight will play a role in the proof that the univalence axiom implies function extensionality.

Theorem 2.7.3. *Suppose that*

$$D : \prod(A,B : \mathsf{Type}),\ (A \simeq B) \to \mathsf{Type}$$

is a dependent type over the equivalences. If there is a function

$$d : \prod(A : \mathsf{Type}),\ D(A,A,\mathrm{idmap}_A),$$

then there is a section $J(D,d) : \prod(A,B : \mathsf{Type})(e : A \simeq B),\ D(A,B,e)$ of D.

PROOF. Suppose that A and B are types and that $e : A \simeq B$. Note that since there is a path

$$\upsilon(A,B)(\upsilon(A,B)^{-1}(e)) \rightsquigarrow e$$

in $A \simeq B$, it suffices to find a term of type $D(A,B,\upsilon(A,B)(\upsilon(A,B)^{-1}(e)))$. From this, it follows that it is enough to find a section of the dependent type D' over the paths of Type given by

$$D'(A,B,p) :\equiv D(A,B,\upsilon(A,B,p))$$

We find such a section with the path induction principle. Note that for an identity path $\mathsf{id}_A : A \rightsquigarrow A$, we have $\upsilon(A,A,\mathsf{id}_A) = \mathrm{idmap}_A$ and hence we get that $D'(A,A,\mathsf{id}_A) = D(A,A,\mathrm{idmap}_A)$. Since we have $d(A) : D(A,A,\mathrm{idmap}_A)$ for each type A, the result follows. ∎

Corollary 2.7.4. *If $A \simeq B$, then $(X \to A) \simeq (X \to B)$ for each space X.*

PROOF. We apply the induction principle for equivalences, so we need to see that the identity equivalence on A induces an equivalence $(X \to A) \simeq (X \to A)$, which is obvious. ∎

Note how we bypass any argument that otherwise would involve the function extensionality principle by using that we can do induction over equivalences. In the rest of this section we will prove the weak function extensionality principle from the univalence axiom, which was one of Voevodsky's first applications of the univalence axiom.

Lemma 2.7.5. *For any type A, the maps*

$$\mathsf{src} :\equiv \lambda w.\, \mathsf{proj}_1\, \mathsf{proj}_1\, w : \Big(\sum(x,y:A),\ x \rightsquigarrow y\Big) \to A$$
$$\mathsf{trg} :\equiv \lambda w.\, \mathsf{proj}_2\, \mathsf{proj}_1\, w : \Big(\sum(x,y:A),\ x \rightsquigarrow y\Big) \to A$$

are both equivalences with inverse

$$\iota :\equiv \lambda x.\langle x,x,\mathsf{id}_x\rangle : A \to \sum(x,y:A),\ x \rightsquigarrow y.$$

PROOF. We will only verify that $\iota \circ \mathsf{src} \sim \mathrm{idmap}$ and that $\mathsf{src} \circ \iota \sim \mathrm{idmap}$. Note that the latter homotopy is trivial. Suppose that $w : \sum(x,y:A),\ x \rightsquigarrow y$. Note that there are $x,y:A$ and $p : x \rightsquigarrow y$ such that $w \rightsquigarrow \langle x,y,p\rangle$, so it suffices to find a path

$$\iota(\mathsf{src}(\langle x,y,p\rangle)) \rightsquigarrow \langle x,y,p\rangle.$$

Now observe that $\iota(\mathsf{src}(\langle x,y,p\rangle)) = \langle x,x,\mathsf{id}_x\rangle$, so it is enough to find a path $\langle x,\mathsf{id}_x\rangle \rightsquigarrow \langle y,p\rangle$. This path is given by the path lifting property, since $p \rightsquigarrow p \cdot \mathsf{id}_x$. The case of trg is similar. ∎

Lemma 2.7.6. *The univalence axiom implies the naive non-dependent function extensionality principle, i.e. that*

$$\prod(A,B:\mathsf{Type})(f,g:A \to B),\ (f \sim g) \to (f \rightsquigarrow g)$$

PROOF. Suppose that $H : f \sim g$ for $f, g : A \to B$. We may construct the functions

$$\tilde{f} := \lambda x.\langle f(x), f(x), \mathsf{id}_{f(x)} \rangle : A \to \sum(u, v : B),\ u \rightsquigarrow v$$
$$\tilde{g} := \lambda x.\langle f(x), g(x), H(x) \rangle : A \to \sum(u, v : B),\ u \rightsquigarrow v.$$

Then $\mathsf{trg} \circ \tilde{f} = \lambda x.f(x) = f$ and $\mathsf{trg} \circ \tilde{g} = \lambda x.g(x) = g$ and hence it is enough to show that there is a path $\tilde{f} \rightsquigarrow \tilde{g}$. Now we use corollary 2.7.4 with the map src, which gives us that

$$\big(A \to \sum(u, v : B),\ u \rightsquigarrow v\big) \simeq (A \to B).$$

Since $\mathsf{src} \circ \tilde{f} = \lambda x.f(x) = \mathsf{src} \circ \tilde{g}$, we get our path $\tilde{f} \rightsquigarrow \tilde{g}$. ∎

Theorem 2.7.7. *The univalence axiom implies the weak function extensionality principle.*

PROOF. Suppose that $P : A \to \mathsf{Type}$ is a dependent type over A and that

$$K : \prod(x : A),\ \mathsf{isContr}(P(x)).$$

Define $U : A \to \mathsf{Type}$ to be the dependent type $\lambda x.\mathsf{unit}$. We will first show that there is a path $P \rightsquigarrow U$. Since both P and U are of type $A \to \mathsf{Type}$, we may apply the above lemma 2.7.6, which gives us that it is enough to verify that $P \sim U$. In other words, we have to prove that $P(x) \rightsquigarrow \mathsf{unit}$ for each $x : A$. Since each $P(x)$ is contractible there is an equivalence $P(x) \simeq \mathsf{unit}$ for each $x : A$. By the univalence axiom we get a path $P(x) \rightsquigarrow \mathsf{unit}$.

Since we have a path $P \rightsquigarrow U$, there is a path between the types

$$\big(\prod(x : A),\ P(x)\big) \rightsquigarrow \big(\prod(x : A),\ U(x)\big)$$

so we only have to verify that $\prod(x : A),\ U(x)$, which is equal to $A \to \mathsf{unit}$, is contractible. Any function from A to unit is homotopic to the function $\lambda x.\mathsf{tt}$, hence the result follows from another application of lemma 2.7.6. ∎

We finish this section with two new applications of the univalence axiom. The first is an observation of Bas Spitters, which he shared with the author at the Fourth Workshop on Formal Topology in Ljubljana in 2012. Like the familiar "families of sets – indexed sets" equivalence, there is an equivalence

$$\big(\sum(X : \mathsf{Type}),\ X \to A\big) \simeq (A \to \mathsf{Type})$$

for any type A. Since we will need paths between types, the proof relies heavily on the univalence axiom. We will also need the following lemma.

Lemma 2.7.8. *Suppose A is a type and consider the dependent type $P : \mathsf{Type} \to \mathsf{Type}$ over Type given by $P(X) := X \to A$. If $e : X \simeq Y$ is an equivalence and $f : X \to A$, then*

$$\upsilon(X, Y)^{-1}(e) \cdot f \rightsquigarrow f \circ e^{-1}.$$

PROOF. The theorem follows by the induction principle for equivalences. Recall that for the identity function idmap_X on X, we have $\upsilon(X,X,\mathrm{idmap}_X) \rightsquigarrow \mathsf{id}_X$ and hence we have $\upsilon(X,X,\mathrm{idmap}_X)\cdot f \rightsquigarrow f$. A path $f \rightsquigarrow f \circ {\mathrm{idmap}_X}^{-1}$ follows from an application of the η-rule. ∎

Theorem 2.7.9 (Spitters). *Assuming the univalence axiom, we have an equivalence*

$$\left(\sum(X : \mathsf{Type}),\ X \to A\right) \simeq (A \to \mathsf{Type})$$

for any type A.

PROOF. Define the function $\varphi : (\sum(X : \mathsf{Type}),\ X \to A) \to (A \to \mathsf{Type})$ by

$$\lambda w.\lambda x.\mathsf{hFiber}(\mathsf{proj}_2\, w, x).$$

In the other direction, define $\psi : (A \to \mathsf{Type}) \to (\sum(x : A),\ X \to A)$ by

$$\lambda P.\langle(\textstyle\sum(x : A),\ P(x)), \mathsf{proj}_1\rangle.$$

To show that $\varphi \circ \psi \sim \mathrm{idmap}$, suppose that $P : A \to \mathsf{Type}$ is a dependent type over A. Then we have

$$\begin{aligned}\varphi(\psi(P)) &= \varphi(\langle(\textstyle\sum(x : A),\ P(x)), \mathsf{proj}_1\rangle)\\ &= \lambda x.\mathsf{hFiber}(\mathsf{proj}_1, x).\end{aligned}$$

By theorem 2.7.7 we have function extensionality and therefore it suffices to show that there is a path

$$\mathsf{hFiber}(\mathsf{proj}_1, x) \rightsquigarrow P(x)$$

for any $x : A$. Another application of the univalence axiom gives us that it suffices to find an equivalence from $\mathsf{hFiber}(\mathsf{proj}_1, x)$ to $P(x)$, which we have done in lemma 2.4.15.

To show that $\psi \circ \varphi \sim \mathrm{idmap}$, we apply the induction principle for dependent sums. Hence it is enough to show that

$$\psi(\varphi(\langle X, f\rangle)) \rightsquigarrow \langle X, f\rangle$$

for each type X and each function $f : X \to A$. Note that

$$\begin{aligned}\psi(\varphi(\langle X, f\rangle)) &= \psi(\lambda x.\mathsf{hFiber}(f, x))\\ &= \langle(\textstyle\sum(x : A),\ \mathsf{hFiber}(f, x)), \mathsf{proj}_1\rangle\end{aligned}$$

Thus, we must find paths

$$\gamma_0 : \left(\sum(x : A),\ \mathsf{hFiber}(f, x)\right) \rightsquigarrow X \qquad \text{and} \qquad \gamma_1 : \gamma_0 \cdot \mathsf{proj}_1 \rightsquigarrow f.$$

By the univalence axiom, we may find γ_0 by constructing an equivalence from $\sum(x:A),\ \mathsf{hFiber}(f,x)$ to X. We do this with the basic equivalences we have seen so far:

$$\begin{aligned}\sum(x:A),\ \mathsf{hFiber}(f,x) &\simeq \sum(x:A)(y:X),\ f(y) \rightsquigarrow x\\ &\simeq \sum(y:X),\ \Big(\sum(x:A),\ f(y) \rightsquigarrow x\Big)\\ &\simeq \sum(y:X),\ \mathsf{unit}\\ &\simeq X.\end{aligned}$$

We have used that the space $\sum(x:A),\ f(y) \rightsquigarrow x$ is contractible for any y. The function we obtain in this way is

$$e := \lambda w.\,\mathsf{proj}_1\,\mathsf{proj}_2\,w \qquad \text{with inverse} \qquad \lambda x.\langle f(x), \langle x, \mathsf{id}_{f(x)}\rangle\rangle.$$

By lemma 2.7.8, we see that

$$\gamma_0 \cdot \mathsf{proj}_1 \rightsquigarrow \mathsf{proj}_1 \circ \gamma_0^{-1} = \lambda x.\,\mathsf{proj}_1\langle f(x), \langle x, \mathsf{id}_{f(x)}\rangle\rangle = \lambda x.f(x) = f,$$

which finishes the proof. ∎

The second application of the univalence axiom asserts that the notions of function and graph coincide. We have chosen to state the theorem in its dependent form, but the proof of the dependent version of the assertion goes along the same lines.

Theorem 2.7.10. *The univalence axiom implies that functions from A to B are total single-valued relations over $A \times B$: i.e. there is an equivalence*

$$\Big(\sum(R:A\to B\to \mathsf{Type})\prod(a:A),\ \mathsf{isContr}\Big(\sum(b:B),\ R(a,b)\Big)\Big) \simeq A \to B.$$

PROOF. For any function $f:A\to B$ there is the relation $R_f : A \to B \to \mathsf{Type}$ given by $R_f(a,b) := f(a) \rightsquigarrow b$. It is immediate from 2.3.16 that $\sum(b:B),\ R_f(a,b)$ is contractible for each $a:A$. This determines a function

$$\psi : (A\to B) \to \sum(R:A\to B\to \mathsf{Type})\prod(a:A),\ \mathsf{isContr}\Big(\sum(b:B),\ R(a,b)\Big).$$

A function φ in the other direction is given by

$$\lambda\langle R,S\rangle.\lambda a.\,\mathsf{proj}_1\,\mathsf{proj}_1\,S(a).$$

If $f:A\to B$, we read from the proof of 2.3.16 that

$$\mathsf{proj}_1(\mathsf{proj}_2\,\psi(f))(a) = \langle f(a), \mathsf{id}_{f(a)}\rangle.$$

Therefore, it is immediate (by extensionality) that $\varphi(\psi(f)) \rightsquigarrow f$ for each $f:A\to B$. To show that $\psi\circ\varphi \sim \mathrm{idmap}$, suppose that $\langle R,S\rangle$ is a total single-valued relation. To find a path from $\psi(\varphi(\langle R,S\rangle))$ to $\langle R,S\rangle$ it suffices to find paths

$$\gamma_0 : \mathsf{proj}_1\,\psi(\varphi(\langle R,S\rangle)) \rightsquigarrow R \qquad \text{and} \qquad \gamma_1 : \gamma_0 \cdot \mathsf{proj}_2\,\psi(\varphi(\langle R,S\rangle)) \rightsquigarrow S.$$

Here we may remark that the space $\prod(a:A),\ \mathsf{isContr}(\sum(b:B),\ R(a,b))$ is itself contractible as a consequence of the weak function extensionality principle and the fact that there is a function $\mathsf{isContr}(X) \to \mathsf{isContr}(\mathsf{isContr}(X))$ for each space X. Hence it suffices to find γ_0. Note that

$$\mathsf{proj}_1\,\psi(\varphi(\langle R,S\rangle))(a,b) = \mathsf{proj}_1\,\mathsf{proj}_1\,S(a) \rightsquigarrow b.$$

Using the function extensionality principle, we have to show that there is a path from the space $R(a,b)$ to the space $\mathsf{proj}_1\,\mathsf{proj}_1\,S(a) \rightsquigarrow b$ in the universe Type, for each $a:A$ and $b:B$. By the univalence axiom, it suffices to show that there is an equivalence

$$R(a,b) \simeq (\mathsf{proj}_1\,\mathsf{proj}_1\,S(a) \rightsquigarrow b).$$

Suppose that $u:R(a,b)$. Then there is the path $(\mathsf{proj}_2\,S(a))(\langle b,u\rangle) : \langle b,u\rangle \rightsquigarrow \mathsf{proj}_1\,S(a)$. The base path of this path is a path from b to $\mathsf{proj}_1\,\mathsf{proj}_1\,S(a)$, which we may invert to obtain the path in the desired direction. Thus, we get the function

$$\tau(a,b) :\equiv \lambda u.\,\mathsf{proj}_1((\mathsf{proj}_2\,S(a))(\langle b,u\rangle))^{-1} : R(a,b) \to (\mathsf{proj}_1\,\mathsf{proj}_1\,S(a) \rightsquigarrow b).$$

Note that both the spaces $\sum(b:B),\ R(a,b)$ and $\sum(b:B),\ \mathsf{proj}_1\,\mathsf{proj}_1\,S(a) \rightsquigarrow b$ are contractible. Therefore, the function $\Sigma_B\tau(a)$ is an equivalence. By theorem 2.4.19 it follows that $\tau(a)$ is a fiberwise equivalence, and hence τ induces our path γ_0. ∎

Corollary 2.7.11. *For any type B, there is an equivalence*

$$B \simeq \sum(P:B\to\mathsf{Type}),\ \mathsf{isContr}\Big(\sum(b:B),\ P(b)\Big).$$

PROOF. Take $A :\equiv \mathsf{unit}$ in the previous theorem. ∎

2.8 A type theoretical Yoneda lemma

In this section we take the point of view that a dependent type P over a type A is the type theoretical analogue of a presheaf over A. This seems not like much, since a dependent type P over A is exactly a function $P:A\to\mathsf{Type}$, but it allows us to take the theory of presheaves as an inspiration for results in homotopy type theory. To manifest this point of view, we denote the transportation $\mathsf{transport}(P,p)$ along a path p in a dependent type P by $P(p)$. The first result from this direction is a type theoretical variant of the Yoneda lemma, stating that the fiber $P(a)$ of P above a is equivalent to the space of all local sections at a of P. In analogy with natural transformations from category theory we make the following definition:

Definition 2.8.1. If P and Q are dependent types over A, we define

$$\mathsf{hom}(P,Q) = \prod(x:A),\ P(x)\to Q(x)$$

of which the terms are called dependent transformations. We will say that a transformation $\sigma:\mathsf{hom}(P,Q)$ is an equivalence if $\sigma(x):P(x)\to Q(x)$ is an equivalence for each $x:A$. ♦

The case where P is the dependent space over A with $P(x) = x \rightsquigarrow a$ for some fixed a in A is of special importance. We will denote this dependent type by $\mathcal{Y}(a)$ to emphasize on the similarity of the role it plays to the Yoneda embedding of a in the category of presheaves over A. Indeed, we will soon be able to prove that $P(a) \simeq \mathsf{hom}(\mathcal{Y}(a), P)$. We call a transformation from $\mathcal{Y}(a)$ to P also a local section of P at a.

Example 2.8.2. If $p : a \rightsquigarrow a'$ is a path in A, then there is the transformation $\mathcal{Y}(p) : \mathsf{hom}(\mathcal{Y}(a), \mathcal{Y}(a'))$ defined by $\lambda x.\lambda q.p \bullet q$. ⋆

Note that we didn't require the naturality in our definition of transformations. This is, however, something that we get for free:

Lemma 2.8.3 (Naturality of dependent transformations). *Suppose that P and Q are dependent types over A and that $\tau : \mathsf{hom}(P,Q)$ is a transformation from P to Q. Then the diagram*

$$\begin{array}{ccc} P(x) & \xrightarrow{\tau(x)} & Q(x) \\ {\scriptstyle P(p)}\downarrow & & \downarrow{\scriptstyle Q(p)} \\ P(y) & \xrightarrow[\tau(y)]{} & Q(y) \end{array}$$

commutes up to homotopy for every path $p : x \rightsquigarrow y$ in A. We define $\tau(p) :\equiv Q(p) \circ \sigma(x)$ when $p : x \rightsquigarrow y$ and $\tau : \mathsf{hom}(P,Q)$.

PROOF. Let $D(x,y,p)$ be the type $Q(p) \circ \sigma(x) \rightsquigarrow \sigma(y) \circ P(p)$. Since $P(\mathsf{id}_x) = \mathsf{id}_{P(x)}$ it follows that $D(x,x,\mathsf{id}_x)$ is inhabited by the canonical term $d(x) :\equiv \mathsf{id}_{\sigma(x)}$. The assertion follows now by the induction principle for identity types. ■

Transformations of dependent types may be composed and just as in the case with natural transformations this is done pointwise:

Definition 2.8.4. Suppose $\tau : \mathsf{hom}(P,Q)$ and $\sigma : \mathsf{hom}(Q,R)$ are transformations of dependent types. Then we define the composition $\sigma \circ \tau : \mathsf{hom}(P,R)$ to be the dependent function $\lambda x.\sigma(x) \circ \tau(x)$. For every dependent type P there is a neutral element with respect to this composition, the identity transformation id_P, which is defined by $\lambda x.\mathsf{id}_{P(x)}$.

For $\tau : \mathsf{hom}(P',P)$ we may furthermore define the precomposition map

$$\mathsf{hom}(\tau, Q) : \mathsf{hom}(P,Q) \to \mathsf{hom}(P',Q)$$

by $\lambda\rho.\rho \circ \tau$ and for $\sigma : \mathsf{hom}(Q,Q')$ we may define the postcomposition map

$$\mathsf{hom}(P, \sigma) : \mathsf{hom}(P,Q) \to \mathsf{hom}(P,Q')$$

by $\lambda\rho.\sigma \circ \rho$. If $\tau : \mathsf{hom}(P',P)$ and $\sigma : \mathsf{hom}(Q,Q')$ are transformations, we define $\mathsf{hom}(\tau,\sigma) : \mathsf{hom}(P,Q) \to \mathsf{hom}(P',Q')$ by $\lambda\rho.\sigma \circ \rho \circ \tau$. It is immediate that there are homotopies

$$\mathsf{hom}(\tau,\sigma) \rightsquigarrow \mathsf{hom}(\sigma,P') \circ \mathsf{hom}(Q,\tau)$$
$$\mathsf{hom}(\tau,\sigma) \rightsquigarrow \mathsf{hom}(Q',\tau) \circ \mathsf{hom}(\sigma,P).$$ ♦

Lemma 2.8.5 (Yoneda lemma). *Assuming function extensionality we have: for any dependent type P over A and any $a : A$ there is an equivalence*

$$\alpha_{P,a} : \mathsf{hom}(\mathcal{Y}(a), P) \simeq P(a)$$

for each $a : A$ and $P : A \to \mathsf{Type}$. The equivalences $\alpha_{P,a}$ are natural in the sense that the diagram

$$\begin{array}{ccc} \mathsf{hom}(\mathcal{Y}(a), P) & \xrightarrow{\alpha_{P,a}} & P(a) \\ {\scriptstyle \mathsf{hom}(\mathcal{Y}(p),\sigma)}\downarrow & & \downarrow{\scriptstyle \sigma(p)} \\ \mathsf{hom}(\mathcal{Y}(a'), P') & \xrightarrow[\alpha_{P',a'}]{} & P'(a') \end{array}$$

commutes for every $p : a \rightsquigarrow a'$ and $\sigma : \mathsf{hom}(P, P')$.

PROOF. Note that if there is a path $p : x \rightsquigarrow y$ in A, then the spaces $P(x)$ and $P(y)$ are equivalent. Therefore, we have

$$\begin{aligned} \mathsf{hom}(\mathcal{Y}(a), P) &= \textstyle\prod(x : A),\ (x \rightsquigarrow a) \to P(x) \\ &\simeq \textstyle\prod(x : A),\ (x \rightsquigarrow a) \to P(a) \\ &\simeq \left(\textstyle\sum(x : A),\ x \rightsquigarrow a\right) \to P(a) \\ &\simeq P(a). \end{aligned}$$

We have also used that $\sum(x : A),\ x \rightsquigarrow a$ is contractible for any $a : A$. Unfolding these equivalences, we see that the function

$$\alpha_{P,a} : \mathsf{hom}(\mathcal{Y}(a), P) \to P(a)$$

is given by $\lambda\tau.\tau(a, \mathsf{id}_a)$, just as in the original Yoneda lemma. For the naturality of α, suppose that we have a path $p : a \rightsquigarrow a'$, a transformation $\sigma : \mathsf{hom}(P, P')$ and a transformation $\tau : \mathsf{hom}(\mathcal{Y}(a), P)$. Then

$$\sigma(p)(\alpha_{P,a}(\tau)) = P'(p)(\sigma(a, \tau(a, \mathsf{id}_a)))$$

and

$$\alpha_{P',a'}(\mathsf{hom}(\mathcal{Y}(p), \sigma)(\tau)) = \sigma(a', \tau(a', p))$$

Induction on p reveals that there is a path between these two terms. ∎

Corollary 2.8.6. *Taking $\mathcal{Y}(b)$ for P we get the equivalence*

$$\mathsf{hom}(\mathcal{Y}(a), \mathcal{Y}(b)) \simeq a \rightsquigarrow b.$$

Remark 2.8.7. We may put some type theoretical notions in a categorical perspective. First, the construction of the homotopy fiber may be recognized as an instance of the comma category construction. When $f : A \to B$ is a function and b is a term of B, we may think of f as a functor from A to B and then the homotopy fiber $\mathsf{hFiber}(f, b)$ is analogue to the category $f{\downarrow}b$.

Recall from the theory of presheaves that there is an equivalence, given by the Yoneda lemma, from the comma category $\mathcal{Y}{\downarrow}P$ of pairs (a, τ), consisting of an object a of A and a natural transformation $\tau : \mathcal{Y}(a) \Rightarrow P$, to the category $\int_A P$ of pairs (a, u) with $u \in P(a)$ for any presheaf P on A. There is an equivalence in the same spirit in type theory, where $\sum(a:A),\ \mathsf{hom}(\mathcal{Y}(a), P)$ plays the role of $\mathcal{Y}{\downarrow}P$ and where $\sum(a:A),\ P(a)$ takes the role of $\int_A P$. ⋆

Corollary 2.8.8. *The total space $\sum(a:A),\ P(a)$ of a dependent space P over A is equivalent to the total space $\sum(a:A),\ \mathsf{hom}(\mathcal{Y}(a), P)$ of of the dependent type of local sections of P.*

2.9 Adjunctions of dependent types

In this section we will introduce the type theoretical variant of adjunctions. The theory we will develop is directly parallel to that of adjunctions of categories, with the main differences being that bijections are replaced by equivalences and diagrams which commute on the nose in category theory are replaced by diagrams which commute up to homotopy. The main goal in this section is to find three familiar adjunctions of presheaf theory:

$$\exists_f \vdash f^*, \qquad f^* \vdash \forall_f \qquad \text{and} \qquad (-) \times P \vdash (-)^P$$

for a function $f : A \to B$ and for a dependent type P. We will develop the theory of type theoretical adjunctions up to the point where we conclude that adjoints are determined up to equivalence.

It should be noted that it is probably desirable to formulate adjunctions in more generality than we do here, using a suitable notion of categories of types, but such a formulation is not entirely clear to the author at the moment of writing this thesis. We stick to a notion of adjunction that meets our current needs.

2.9.1 Definition and basic properties of adjunctions

We will use the function extensionality principle throughout this section; often we will do so without explicit mentioning. Since we will investigate functions from $A \to \mathsf{Type}$ to $B \to \mathsf{Type}$, it will be useful to start with introducing some notation.

Definition 2.9.1. We denote the type $A \to \mathsf{Type}$ of dependent types over A by $\hat{A}$. ⧫

Definition 2.9.2. Suppose that A and B are types. A functor of dependent types is a pair $\langle F_0, F_1 \rangle$ consisting of a function

$$F_0 : \hat{A} \to \hat{B}$$

transforming the dependent types over A to dependent types over B, and a function

$$F_1 : \prod\{P,P' : A \to \mathsf{Type}\},\ \mathsf{hom}(P,P') \to \mathsf{hom}(F_0(P),F_0(P'))$$

such that F_1 preserves idmap_P and composition: there are paths

$$F_1(\mathrm{idmap}_P) \rightsquigarrow \mathrm{idmap}_{F_0(P)} \qquad \text{and} \qquad F_1(\sigma \circ \tau) \rightsquigarrow F_1(\sigma) \circ F_1(\tau)$$

for each $P,P',P'' : A \to \mathsf{Type}$ and each $\tau : \mathsf{hom}(P,P')$ and $\sigma : \mathsf{hom}(P',P'')$. Often, we will not distinguish between F_0 and F_1 in our notation and we will simply write F. ⧫

Definition 2.9.3. If $F,G : \hat{A} \to \hat{B}$ are two functors, a natural transformation from F to G is a dependent function μ of type $\prod(P:\hat{A}),\ \mathsf{hom}(FP,GP)$ with the property that the square

$$\begin{array}{ccc} FQ & \xrightarrow{\mu_Q} & GQ \\ {\scriptstyle F(\tau)}\uparrow & & \uparrow{\scriptstyle G(\tau)} \\ FP & \xrightarrow[\mu_P]{} & GP \end{array}$$

commutes up to homotopy for each morphism $\tau : \mathsf{hom}(P,Q)$ of dependent types over A. The space of natural transformations from F to G is denoted by $F \Rightarrow G$. ⧫

Definition 2.9.4. For two operations $F : \hat{A} \to \hat{B}$ and $G : \hat{B} \to \hat{A}$ we say that F is left adjoint to G if there are equivalences

$$\Theta(P,Q) : \mathsf{hom}(F(P),Q) \simeq \mathsf{hom}(P,G(Q))$$

for every pair of dependent types $P : \hat{A}$ and $Q : \hat{B}$, which are natural in the sense that the diagram

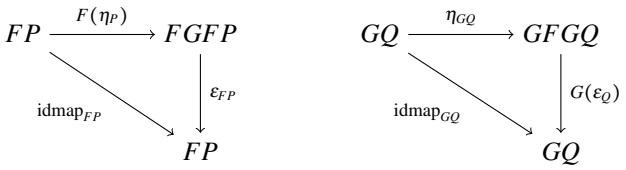
$$\begin{array}{ccc} \mathsf{hom}(F(P),Q) & \xrightarrow{\simeq} & \mathsf{hom}(P,G(Q)) \\ {\scriptstyle \mathsf{hom}(F(\tau),\sigma)}\downarrow & & \downarrow{\scriptstyle \mathsf{hom}(\tau,G(\sigma))} \\ \mathsf{hom}(F(P'),Q') & \xrightarrow[\simeq]{} & \mathsf{hom}(P',G(Q')) \end{array}$$

commutes up to a homotopy $\Psi(\tau,\sigma)$ for every $\tau : \mathsf{hom}(P,P')$ and $\sigma : \mathsf{hom}(Q,Q')$. We write $F \vdash G$ for the space of such pairs $\langle \Theta, \Psi \rangle$. ⧫

Lemma 2.9.5. *For every adjunction $F \vdash G$, where $F : \hat{A} \to \hat{B}$ and $G : \hat{B} \to \hat{A}$, there are natural transformations*

$$\eta : \mathsf{id}_{\hat{A}} \Rightarrow GF$$
$$\varepsilon : FG \Rightarrow \mathsf{id}_{\hat{B}}.$$

with the property that the triangles



commute up to homotopy for each $P : \hat{A}$ and each $Q : \hat{B}$. The function η is said to be the unit of the adjunction and ε is said to be the counit of the adjunction.

PROOF. To define η, note that we have an equivalence

$$\Theta_{P,FP} : \mathsf{hom}(FP, FP) \simeq \mathsf{hom}(P, GFP)$$

for each $P : A \to \mathsf{Type}$, so we may define $\eta_P :\equiv \Theta_{P,FP}(\mathsf{id}_{FP})$. To see that η is a natural transformation, note that we have the paths

$$GF(\tau) \circ \Theta_{P,FP}(\mathsf{id}_{FP}) \rightsquigarrow \Theta_{P,FP'}(F(\tau)) \rightsquigarrow \Theta_{P',FP'}(\mathsf{id}_{FP'}) \circ \tau$$

from the naturality of Θ. The natural transformation ε is defined similarly, using the equivalence

$$\Theta_{GQ,Q} : \mathsf{hom}(FGQ, Q) \simeq \mathsf{hom}(GQ, GQ).$$

To prove the commutativity of the triangle on the left, we use the naturality once more:

$$\begin{aligned} \varepsilon_{FP} \circ F(\eta_P) &= \Theta^{-1}(\mathsf{id}_{GFP}) \circ F(\Theta(\mathsf{id}_{FP})) \\ &\rightsquigarrow \Theta^{-1}(\mathsf{id}_{GFP} \circ \Theta(\mathsf{id}_{FP})) \\ &\rightsquigarrow \Theta^{-1}(\Theta(\mathsf{id}_{FP})) \\ &\rightsquigarrow \mathsf{id}_{FP}. \end{aligned}$$

The other triangle follows by a similar calculation. ∎

Lemma 2.9.6. *Suppose that $F : \hat{A} \to \hat{B}$ has right adjoints $G, G' : \hat{B} \to \hat{A}$. Then there is a natural transformation $\mu : G \Rightarrow G'$ such that each $\mu_Q : G(Q) \to G'(Q)$ is an equivalence.*

PROOF. Suppose that $Q : B \to \mathsf{Type}$ and that $a : A$. Then we have, by the Yoneda lemma, the equivalences

$$G(Q)(a) \simeq \mathsf{hom}(\mathcal{Y}(a), G(Q)) \simeq \mathsf{hom}(F(\mathcal{Y}(a)), Q) \simeq \mathsf{hom}(\mathcal{Y}(a), G'(Q)) \simeq G'(Q)(a).$$

The naturality follows from the naturality of the Yoneda lemma. ∎

Lemma 2.9.7. *Suppose that $G : \hat{B} \to \hat{A}$ has left adjoints $F, F' : \hat{A} \to \hat{B}$. Then there is a natural transformation $\mu : F \Rightarrow F'$ such that each $\mu_P : F(P) \to F'(P)$ is an equivalence.*

PROOF. Suppose that $\langle \Theta, \Psi \rangle : F \vdash G$ and that $\langle \Theta', \Psi' \rangle : F' \vdash G$. Then we have the equivalence

$$\Theta_{P,F'P} : \mathsf{hom}(FP, F'P) \simeq \mathsf{hom}(P, GF'P)$$

and hence we find $\Theta^{-1}(\eta'_P) : \mathsf{hom}(FP, F'P)$, where η' is the unit of the adjunction $F' \vdash G$. Similarly, we find $\Theta'^{-1}(\eta_P) : \mathsf{hom}(F'P, FP)$. To prove that $\Theta'^{-1}(\eta_P) \circ \Theta^{-1}(\eta'_P) \rightsquigarrow \mathsf{id}_{FP}$, note that it suffices to show that $\Theta(\Theta'^{-1}(\eta_P) \circ \Theta^{-1}(\eta'_P)) \rightsquigarrow \eta_P$. This is just the same calculation as in the categorical argument:

$$\begin{aligned}
\Theta(\Theta'^{-1}(\eta_P) \circ \Theta^{-1}(\eta'_P)) \rightsquigarrow \eta_P &\rightsquigarrow G\Theta'^{-1}(\eta_P) \circ \eta'_P \\
&= G\Theta'^{-1}(\eta_P) \circ \Theta'(\mathsf{id}_{F'P}) \\
&\rightsquigarrow \Theta'(\Theta'^{-1}(\eta_P) \circ \mathsf{id}_{F'P}) \\
&\rightsquigarrow \eta_P
\end{aligned}$$

The proof that there is a homotopy $\Theta^{-1}(\eta') \circ \Theta'^{-1}(\eta_P) \sim \mathsf{id}$ is similar. The naturality of the maps $\lambda P.\Theta^{-1}(\eta'_P)$ and $\lambda P.\Theta'^{-1}(\eta_P)$ follows from the naturality of Θ, Θ', η and η'. ∎

Corollary 2.9.8. *Suppose that $F, G : (B \to \mathsf{Type}) \to (A \to \mathsf{Type})$ have right adjoints R_F and R_G respectively. If*

$$\prod(b : B),\ F(\mathcal{Y}(b)) \simeq G(\mathcal{Y}(b))$$

then $F(Q) \simeq G(Q)$ for every $Q : B \to \mathsf{Type}$.

PROOF. We will show that $R_F \simeq R_G$. It is then an immediate consequence that $F \simeq G$. Suppose that $P : A \to \mathsf{Type}$ and that $b : B$. Then we have

$$R_F(P)(b) \simeq \mathsf{hom}(\mathcal{Y}(b), R_F(P)) \simeq \mathsf{hom}(F(\mathcal{Y}(b)), P) \simeq \mathsf{hom}(G(\mathcal{Y}(b)), P) \simeq R_G(P)(b).$$

∎

2.9.2 Existential and universal quantification of dependent types

We will now begin with proving the three adjunctions we stated at the beginning of this section. The arguments will be more type theoretical from now on.

Definition 2.9.9. Suppose that $f : A \to B$ is a function. Then we define the functor $f^* : \hat{B} \to \hat{A}$ by substitution along f, i.e. $f^*(Q)(x) :\equiv Q(f(x))$ for $x : A$ and $Q : \hat{B}$. If $\tau : \mathsf{hom}(Q, Q')$ is a transformation of dependent types, we define

$$f^*(\tau) :\equiv \lambda x.\lambda u.\tau(f(x), u).$$

It is immediate that f^* defines a functor. ♦

Definition 2.9.10. Suppose that $f : A \to B$ is a function. Then we define the functor $\exists_f : \hat{A} \to \hat{B}$ by

$$\exists_f(P)(b) \equiv \sum(x : A),\ (f(x) \rightsquigarrow b) \times P(x).$$

For $\tau : \mathsf{hom}(P, P')$ we define

$$\exists_f(\tau) := \lambda b.\lambda\langle x, p, u\rangle.\langle x, p, \tau(x, u)\rangle. \qquad \blacklozenge$$

Remark 2.9.11. Consider the projection $\mathsf{proj}_1 : B \times Y \to B$. Then

$$\begin{aligned}
\exists_{\mathsf{proj}_1}(P)(b) &= \textstyle\sum(\langle b', y\rangle : B \times Y),\ (b' \rightsquigarrow b) \times P(b', y) \\
&\simeq \textstyle\sum(b' : B)(y : Y),\ (b' \rightsquigarrow b) \times P(b, y) \\
&\simeq \textstyle\sum(y : Y),\ P(b, y) \times \left(\sum(b' : B),\ b' \rightsquigarrow b\right) \\
&\simeq \textstyle\sum(y : Y),\ P(\langle b, y\rangle).
\end{aligned}$$

We have used the result from lemma 2.3.16 that the space $\sum(b' : B),\ b' \rightsquigarrow b$ is contractible. What this shows is that $\exists_f$ is equivalent to our original dependent sum when f is a projection. $\star$

Lemma 2.9.12. *The functor $\exists_f$ is left adjoint of the functor f^*.*

PROOF. Suppose that $P : \hat{A}$ and that $Q : \hat{B}$. We will first show that

$$\Theta : \mathsf{hom}(\exists_f(P), Q) \simeq \mathsf{hom}(P, f^*(Q)).$$

The proof uses the equivalence of lemma 2.6.3 and the Yoneda lemma 2.8.5.

$$\begin{aligned}
\mathsf{hom}(\exists_f(P), Q) &= \textstyle\prod(b : B),\ \left(\sum(a : A),\ (f(a) \rightsquigarrow b) \times P(a)\right) \to Q(b) \\
&\simeq \textstyle\prod(b : B)(a : A),\ ((f(a) \rightsquigarrow b) \times P(a)) \to Q(b) \\
&\simeq \textstyle\prod(a : A),\ P(a) \to \prod(b : B),\ (b \rightsquigarrow f(a)) \to Q(b) \\
&\simeq \textstyle\prod(a : A),\ P(a) \to Q(f(a)) \\
&= \mathsf{hom}(P, f^*(Q)).
\end{aligned}$$

By unfolding the definitions of the equivalences involved, we see that the equivalence we constructed is given by

$$\lambda\mu.\lambda a, u.\mu(f(a), \langle a, \mathsf{id}_{f(a)}, u\rangle).$$

Using this presentation of Θ, verifying its naturality is a short but a bit messy calculation. $\blacksquare$

Definition 2.9.13. Suppose that $f : A \to B$ is a function. Define the functor $\forall_f : \hat{A} \to \hat{B}$ by

$$\forall_f(P)(b) = \textstyle\prod(a : A),\ (f(a) \rightsquigarrow b) \to P(a)$$

On transformations $\tau : \mathsf{hom}(P, P')$, we define $\forall_f$ by

$$\forall_f(\tau) := \lambda b, s.\tau \circ s. \qquad \blacklozenge$$

Remark 2.9.14. Again, we may consider the map $\mathsf{proj}_1 : B \times Y \to B$ to verify that $\forall_{\mathsf{proj}_1}(P)(b) \simeq \prod(y:Y),\ P(b,y)$.

$$\begin{aligned}\forall_{\mathsf{proj}_1}(P)(b) &= \prod(\langle b',y\rangle : B\times Y),\ (b' \rightsquigarrow b) \to P(b',y)\\ &\simeq \prod(b':B)(y:Y),\ (b' \rightsquigarrow b) \to P(b,y)\\ &\simeq \Big(\sum(b':B),\ b' \rightsquigarrow b\Big) \to \prod(y:Y),\ P(b,y)\\ &\simeq \prod(y:Y),\ P(b,y).\end{aligned}$$

Thus, $\forall_f$ generalizes the original dependent product construction. ⋆

Lemma 2.9.15. *For all $P : A \to \mathsf{Type}$ and $Q : B \to \mathsf{Type}$ we have the equivalence*

$$\mathsf{hom}(f^*(Q),P) \simeq \mathsf{hom}(Q,\forall_f(P)).$$

PROOF. Suppose that $P : \hat{A}$ and that $Q : \hat{B}$. We will first show that we have an equivalence

$$\Theta : \mathsf{hom}(Q,\forall_f(P)) \simeq \mathsf{hom}(f^*(Q),P).$$

As in the proof of the $\exists_f \vdash f^*$ adjunction, we use several known equivalences:

$$\begin{aligned}\mathsf{hom}(Q,\forall_f(P)) &= \prod(b:B),\ Q(b) \to \prod(a:A),\ (f(a) \rightsquigarrow b) \to P(a)\\ &\simeq \prod(a:A)(b:B),\ Q(b) \to (f(a) \rightsquigarrow b) \to P(a)\\ &\simeq \prod(a:A),\ \Big(\sum(b:B),\ Q(b)\times(f(a) \rightsquigarrow b)\Big) \to P(a)\\ &\simeq \prod(a:A),\ Q(f(a)) \times \Big(\sum(b:B),\ f(a) \rightsquigarrow b\Big) \to P(a)\\ &\simeq \prod(a:A),\ Q(f(a)) \to P(a)\\ &= \mathsf{hom}(f^*(Q),P).\end{aligned}$$

Unfolding the definitions of the equivalences that we have used here, we get that Θ is the function

$$\lambda\mu.\lambda a,u.\mu(f(a),u)(a,\mathsf{id}_{f(a)}),$$

which is natural by a messy but not complicated calculation. ■

2.9.3 Exponentiation of dependent types

Definition 2.9.16. Suppose that P is a dependent type over A. We define the functor $(-)\times P : \hat{A} \to \hat{A}$ by

$$(Q\times P)(x) :\equiv P'(x)\times P(x).$$

If $\tau : \mathsf{hom}(Q,R)$ is a transformation of dependent types, we define $\tau\times P : \mathsf{hom}(Q\times P, R\times P)$ by

$$\lambda x,\langle u,v\rangle.\langle\tau(x,u),v\rangle.$$ ♦

Definition 2.9.17. Suppose that P is a dependent type over A. We define the functor $(-)^P : \hat{A} \to \hat{A}$ by

$$Q^P(a) = \mathsf{hom}(\mathcal{Y}(a) \times P, Q).$$

If $\tau : \mathsf{hom}(Q,R)$ is a transformation of dependent types, we define $\tau^P : \mathsf{hom}(Q^P, R^P)$ by

$$\lambda x, f.\tau \circ f.$$

Since $(-)^P$ is defined by post-composition, it is immediate that it is functorial. ♦

Lemma 2.9.18. *The functor $(-)^P$ is right adjoint to the functor $(-) \times P$.*

PROOF. Suppose that $Q, R : \hat{A}$. We will first show that there is an equivalence

$$\Theta : \mathsf{hom}(R \times P, Q) \simeq \mathsf{hom}(R, Q^P).$$

Note that if there is a path from y to x, then the spaces $P(y)$ and $P(x)$ are equivalent. Thus we can make the following calculation using basic equivalences:

$$\begin{aligned}
\mathsf{hom}(R, Q^P) &= \textstyle\prod(x:A),\ R(x) \to \mathsf{hom}(\mathcal{Y}(x) \times P, Q) \\
&= \textstyle\prod(x:A),\ R(x) \to \left(\prod(y:A),\ (y \rightsquigarrow x) \times P(y) \to Q(y)\right) \\
&\simeq \textstyle\prod(x:A),\ R(x) \to \left(\prod(y:A),\ (y \rightsquigarrow x) \times P(x) \to Q(y)\right) \\
&\simeq \textstyle\prod(x:A),\ R(x) \to P(x) \to \left(\prod(y:A),\ (y \rightsquigarrow x) \to Q(y)\right) \\
&\simeq \textstyle\prod(x:A),\ R(x) \times P(x) \to Q(x) \\
&= \mathsf{hom}(R \times P, Q).
\end{aligned}$$

Unfolding this equivalence, we get

$$\lambda\mu.\lambda x, \langle v, u\rangle.\mu(x, v, x, \mathsf{id}_x, u)$$

which can seen to be natural by a calculation only involving folding and unfolding the definitions. ■

Remark 2.9.19. That the categorical definition of exponentiation doesn't fail is no surprise. But actually we could get exponentiation for free by defining $P \to Q$ to be the dependent type over A given by $(P \to Q)(a) :\equiv P(a) \to Q(a)$. The correspondence

$$\mathsf{hom}(R, P \to Q) \simeq \mathsf{hom}(R \times P, Q)$$

is then immediate. By taking R to be $\mathcal{Y}(a)$ and using the Yoneda lemma, we get

$$(P \to Q)(a) \simeq \mathsf{hom}(\mathcal{Y}(a), P \to Q) \simeq \mathsf{hom}(\mathcal{Y}(a) \times P, Q) = Q^P(a).$$

Hence the dependent types $P \to Q$ and Q^P are equivalent, which comes down to the fact that we could have gotten away with the naive approach regarding exponentiation. ⋆

3. HIGHER INDUCTIVE TYPES

In this chapter we begin with our investigation of higher inductive types. Our approach has a similar spirit to that of ordinary inductive types: we define a type by specifying its points and paths and then state the induction principle as a minimality principle. And also in the case of higher inductive types the induction principle states that there is a section of every dependent type that resembles the structure of the basic constructors.

It is rather simple to introduce the basic constructors in such a way that you get the space you want. In the case of the interval you take two points and connect them with a path. In the case of the circle you take one point and a path that begins and ends in that point. The harder part is to define the induction principle. We will explain how to get the induction principle when the constructors are easy but reasonably general but we do have a general principle yet to define all the higher inductive types. Therefore it becomes important that we show that the types we have defined inductively are indeed determined uniquely up to equivalence. We prove this fact by proving a correspondence theorem for each of the higher inductive types we define: the space of sections of a dependent type over the higher inductive type is equivalent to the space of data which is needed to apply the induction principle for constructing a section. This correspondence theorem is a dependent version of the statement that the higher inductive type is initial in a certain sense. From this point of view, higher inductive types are like colimits.

3.1 The idea of higher inductive types

The inductive types we have seen so far have all in common that they are given by specifying their points and the property that towards everything else which also models this structure of points there is a function from the inductively defined space. This idea allowed us to construct a great deal of spaces, but with identity types around the story doesn't end here. What if we can also specify certain paths of a space. And paths between paths, i.e. 2-cells. And even higher cells, or maybe cells at each level. It would, for example, allow us to define an interval or a circle, a sphere, a torus, a Klein bottle and many more spaces. The circle could be given as a type which consists of a point and a (non-identity) path from that point to itself. The sphere would consist of a point and a 2-cell from the identity path on that point to itself. The difficulty with higher inductive spaces is not to give the basic constructors, i.e. the points and the paths. The difficulty is rather to formulate the induction principle which it has to satisfy. In this section we will do the preliminary work that allows us to state the induction principles for inductive spaces with paths as basic constructors and work with them.

In dependent type theory, induction principles are formulated as the existence of a section of a dependent type when that dependent type resembles the structure of the defined space. In the case of higher inductive spaces, the idea is exactly the same. Suppose that a space A has a path $p : x \rightsquigarrow y$ and that $f : \prod(x : A),\ P(x)$ is a section of a dependent type over A. Then there is the path $f(p) : p \cdot f(x) \rightsquigarrow y$ in $P(y)$. Thus, if

we want to define a section f of P with the property $f(x) \rightsquigarrow u$ and $f(y) \rightsquigarrow v$ for some $u : P(x)$ and $v : P(y)$, then there should be a path from $p \cdot u \rightsquigarrow v$.

Lemma 3.1.1. *Suppose that $f, g : \prod(x : A),\ P(x)$ are sections of a dependent type P over A and consider the dependent type Q over A given by*

$$Q(x) :\equiv f(x) \rightsquigarrow g(x),$$

so that the sections of Q are homotopies from f to g. The square

$$\begin{array}{ccc} f(y) & \overset{Q(p)(q)}{\rightsquigarrow} & g(y) \\ {\scriptstyle f(p)^{-1}}\big\downarrow & & \big\uparrow{\scriptstyle g(p)} \\ p \cdot f(x) & \underset{P(p)(q)}{\rightsquigarrow} & p \cdot g(x) \end{array}$$

commutes for every path $p : x \rightsquigarrow y$ in A and every $q : Q(x)$.

PROOF. This is immediate with induction on p. ∎

Lemma 3.1.2. *Suppose that P is a dependent space over A, that $p : x \rightsquigarrow y$ is a path in A and consider the dependent type Q over $P(x) \times P(y)$ given by $Q(a, b) :\equiv p \cdot a \rightsquigarrow b$. For $\alpha : u \rightsquigarrow u'$ in $P(x)$ and $\beta : v \rightsquigarrow v'$ in $P(y)$ we have*

$$\alpha \cdot q \rightsquigarrow q \bullet P(p)(\alpha)^{-1}$$
$$\beta \cdot q \rightsquigarrow \beta \bullet q$$

for each $q : Q(u, v)$. In the case that Q is the dependent type over $P(x)$ given by $Q(u) :\equiv p \cdot u \rightsquigarrow u$, we have

$$\alpha \cdot q \rightsquigarrow \alpha \bullet q \bullet P(p)(\alpha)^{-1}$$

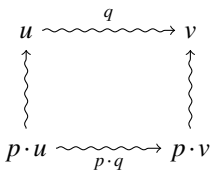
for $\alpha : u \rightsquigarrow u'$ and $q : Q(u)$.

PROOF. All the paths can be found with double induction. ∎

All the induction principle will be stated in a dependent form, but it will be useful to state (and prove) a non-dependent version as well. The following lemma is our main tool to pass from a dependent induction principle to a non-dependent one. It asserts that if P is a constant dependent type, if $P = \lambda x.B$ for some type B, then transportation doesn't really do something.

Lemma 3.1.3. *Suppose that A and B are types and consider the constant dependent type $P :\equiv \lambda x.B$ over A. Then there is a path $u \rightsquigarrow p \cdot u$ in B for any term $u : B$ and any*

path $p : x \rightsquigarrow y$ in A. Moreover, if $q : u \rightsquigarrow v$ is a path in B, then the square



commutes.

PROOF. Immediate with induction over p. ∎

3.2 Examples of some inductive types with paths

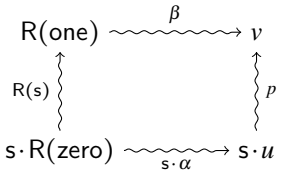
With the ideas that were presented as a motivation for the inductive definition of the circle, we may continue to define other higher inductive types as well. We will cook up a definition of the interval, the real line, the 2-sphere and the torus. We begin with the interval.

3.2.1 The interval

Definition 3.2.1. The interval I is defined as an inductive space with basic constructors $\mathsf{zero}, \mathsf{one} : \mathsf{I}$ and $\mathsf{s} : \mathsf{zero} \rightsquigarrow \mathsf{one}$. The induction principle for the interval is that whenever we have

$$P : \mathsf{I} \to \mathsf{Type} \qquad v : P(\mathsf{one})$$
$$u : P(\mathsf{zero}) \qquad p : \mathsf{s} \cdot u \rightsquigarrow v$$

then there is a section $\mathsf{R} : \prod(t : \mathsf{I}),\ P(x)$ of P with the property that there are paths $\alpha : \mathsf{R}(\mathsf{zero}) \rightsquigarrow u$ and $\beta : \mathsf{R}(\mathsf{one}) \rightsquigarrow v$ and a path γ witnessing that the square

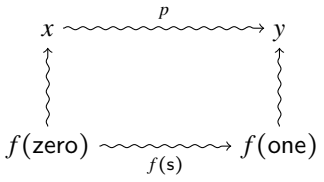


commutes. ♦

There is of course also a non-dependent version of the induction principle.

Lemma 3.2.2. *If A is a type and $p : x \rightsquigarrow y$ is a path in A, then there is a function*

$f : I \to A$ with the property that $f(\mathsf{zero}) \rightsquigarrow x$, $f(\mathsf{one}) \rightsquigarrow y$ and such that the square

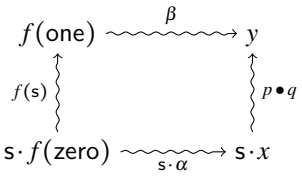


commutes.

PROOF. Suppose we have a type A and a path $p : x \rightsquigarrow y$ in A. Let P be the dependent type over I given by $\lambda t.A$. Then we have $x : P(\mathsf{zero})$ and $y : P(\mathsf{one})$. Since there is a canonical path $q : \mathsf{s} \cdot x \rightsquigarrow x$ in $P(y)$, we get a path $\mathsf{s} \cdot x \rightsquigarrow y$. The induction principle for the interval now gives us a function $f : \mathsf{I} \to A$ for which there are paths

$$\alpha : \mathsf{s} \cdot f(\mathsf{zero}) \rightsquigarrow x \qquad \text{and} \qquad \beta : \mathsf{s} \cdot f(\mathsf{one}) \rightsquigarrow y$$

and a witness γ of the commutativity of the square



The commutativity of the diagram in the statement now follows from lemma 3.1.3. ∎

Theorem 3.2.3 (Correspondence theorem). *For any dependent space P over the interval, the space of sections of P is equivalent to the space*

$$\mathsf{Constr}(P) :\equiv \sum (u : P(\mathsf{zero}))(v : P(\mathsf{one})),\ \mathsf{s} \cdot u \rightsquigarrow v$$

of initial data for the induction principle of the interval.

PROOF. Suppose that P is a dependent space over the interval. Every section f of P determines the triple $\langle f(\mathsf{zero}), f(\mathsf{one}), f(\mathsf{s}) \rangle$ of $\mathsf{Constr}(P)$ and the induction principle defines a map from $\mathsf{Constr}(P)$ to the space of sections of P. Denote these maps by φ and ψ respectively. We have to verify that $\psi(\varphi(f)) \rightsquigarrow f$ for every section f of P and that $\varphi(\psi(u, v, p)) \rightsquigarrow \langle u, v, p \rangle$ for every triple $\langle u, v, p \rangle$ in $\mathsf{Constr}(P)$.

First we check that $\psi(\varphi(f)) \rightsquigarrow f$ and we denote $\psi(\varphi(f))$ by $\tilde{f}$. Thus, we need to find a term of the space

$$\prod (t : \mathsf{I}),\ \tilde{f}(t) \rightsquigarrow f(t).$$

We will do this using the induction principle for the interval, using the dependent type Q over I given by $Q(t) :\equiv \tilde{f}(t) \rightsquigarrow f(t)$. A path α from $\tilde{f}(\mathsf{zero}) \rightsquigarrow f(\mathsf{zero})$ is given by

the induction principle (remember that $\tilde{f}$ is defined by the induction principle), just like a path β from $\tilde{f}(\mathsf{one})$ to $f(\mathsf{one})$. It is left to find a path from $\mathsf{s} \cdot \alpha = Q(\mathsf{s})(\alpha)$ to β. From lemma 3.1.1 we know that there is a path

$$Q(\mathsf{s})(\alpha) \rightsquigarrow f(\mathsf{s}) \bullet P(\mathsf{s})(\alpha) \bullet \tilde{f}(\mathsf{s})^{-1}$$

A path from the latter to β is then given by a modification of γ. We have gathered all the information necessary to apply the induction principle, so we conclude that there is a path $\tilde{f}(x) \rightsquigarrow f(x)$ for each x in the interval, and hence that $\psi \circ \varphi \sim \mathrm{idmap}$.

For the path $\varphi \circ \psi \sim \mathrm{idmap}$, i.e. that $\varphi(\psi(w)) \rightsquigarrow w$ for each $w : \mathsf{Constr}(P)$. Note that, by the induction principle for dependent sums, it suffices to show that

$$\varphi(\psi(\langle u, v, p\rangle)) \rightsquigarrow \langle u, v, p\rangle$$

for each $u : P(\mathsf{zero})$, $v : P(\mathsf{one})$ and that $p : \mathsf{s} \cdot u \rightsquigarrow v$. Since $R :\equiv \varphi(\langle u, v, p\rangle)$ is defined by the induction principle, there are paths

$$\alpha : R(\mathsf{zero}) \rightsquigarrow u, \qquad \beta : R(\mathsf{one}) \rightsquigarrow v \qquad \text{and} \qquad \gamma : \beta \bullet R(\mathsf{s}) \bullet (\mathsf{s} \cdot \alpha)^{-1} \rightsquigarrow p.$$

To show that $\varphi(R) \rightsquigarrow \langle u, v, p\rangle$, it is only left to show that there is a path $\beta \cdot (\alpha \cdot R(\mathsf{s})) \rightsquigarrow p$. Such a path is constructed from γ and an application of lemma 3.1.2. ∎

Corollary 3.2.4. *Suppose that A is a type. Then the space of functions $\mathsf{I} \to A$ is equivalent to the space*

$$\sum(x, y : A),\ x \rightsquigarrow y$$

Recall that we have already seen in lemma 2.7.5 that the spaces A and $\sum(x, y : A),\ x \rightsquigarrow y$ are equivalent. So the above corollary has the consequence that $(\mathsf{I} \to A) \simeq A$. Such an equivalence holds also for unit, which is a hint that the interval is contractible.

Lemma 3.2.5. *The interval is contractible.*

PROOF. Consider the dependent type P over the interval given by $P(x) = x \rightsquigarrow \mathsf{one}$. Then $\mathsf{s} : P(\mathsf{zero})$ and $\mathsf{id}_{\mathsf{one}} : P(\mathsf{one})$. The lemma is proven once we have found a section of P. To apply the induction principle for the interval in order to show the existence of such a path, we need a path from $\mathsf{s} \cdot \mathsf{s}$ to $\mathsf{id}_{\mathsf{one}}$ in $P(\mathsf{one})$. This path is given by lemma 2.2.9. ∎

Since the interval is contractible, introducing it does not bring a lot of new things to type theory. But that changes if we introduce a variant of the interval where the conversion rules are stated in a definitional form. An observation of Mike Shulman[2] has revealed that a definitional interval implies the naive function extensionality principle.

Definition 3.2.6. We define the type $\mathbf{I}$ inductively with basic constructors

$$\mathbf{zero},\ \mathbf{one} : \mathbf{I} \qquad \text{and} \qquad \mathbf{s} : \mathbf{zero} \rightsquigarrow \mathbf{one}.$$

[2] http://homotopytypetheory.org/2011/04/04/an-interval-type-implies-function-extensionality/

The induction principle for $\mathbf{I}$ is that for every dependent type P over $\mathbf{I}$ for which there are

$$u : P(\mathbf{zero}), \qquad v : P(\mathbf{one}) \qquad \text{and} \qquad p : \mathbf{s} \cdot u \rightsquigarrow v,$$

there is a section $\mathbf{R} : \prod(t : \mathbf{I}),\ P(t)$ with the property that

$$\mathbf{R}(\mathbf{zero}) = u, \qquad \mathbf{R}(\mathbf{one}) = v \qquad \text{and} \qquad \mathbf{R}(\mathbf{s}) = p. \qquad \blacklozenge$$

We state the non-dependent induction principle without proof. The proof is similar as in the propositional case, but without the complications that came from the fact that the computation rule was intensional instead of definitional.

Lemma 3.2.7. *For every type A and every path $p : x \rightsquigarrow y$ in A there is a function $f : \mathbf{I} \to A$ with the property that $f(\mathbf{zero}) = x$, $f(\mathbf{one}) = y$ and $f(\mathbf{s}) = p$.*

Theorem 3.2.8. *Suppose that $f, g : \prod(x : A),\ P(x)$ are sections of a dependent type P over A. There is a function*

$$(f \sim g) \to (f \rightsquigarrow g).$$

PROOF. For each $x : A$, there is a function $m_x : (f(x) \rightsquigarrow g(x)) \to \mathbf{I} \to P(x)$ with the property that $m_x(p, \mathbf{s}) = p$ for each $p : f(x) \rightsquigarrow f(y)$. Hence there is a function

$$m : (f \sim g) \to \prod(x : A),\ \mathbf{I} \to P(x).$$

Moreover, we have

$$k := \lambda\varphi.\lambda t.\lambda x.\varphi(x, t) : \Big(\prod(x : A),\ \mathbf{I} \to P(x)\Big) \to \Big(\mathbf{I} \to \prod(x : A),\ P(x)\Big)$$

Then $k(m(H))(\mathbf{s})$ is a path from f to g for any homotopy $H : f \sim g$. ∎

3.2.2 The circle

We also present the circle as a higher inductive space, following the same ideas with which we formulated the interval as an inductive space. The idea of defining the circle as an inductive space has been considered at the Oberwolfach workshop in 2011 on homotopy type theory and has been written down first by Peter Lumsdaine on the homotopy type theory blog[3].

As we have done with the interval, we will show a correspondence theorem for the circle and derive it's uniqueness from it. Also, we will show that the circle is contractible if and only if for any type A and any two paths $p, q : x \rightsquigarrow y$ in A there is a path $s : p \rightsquigarrow q$. This last condition is known as the principle of *Uniqueness of identity proofs*.

[3] http://homotopytypetheory.org/2011/04/24/higher-inductive-types-a-tour-of-the-menagerie/

Definition 3.2.9. We define the type circle inductively with basic constructors

$$\mathsf{base} : \mathsf{circle} \qquad \text{and} \qquad \mathsf{loop} : \mathsf{base} \rightsquigarrow \mathsf{base}.$$

The induction principle for the circle is that whenever P is a dependent type over circle for which there are

$$u : P(\mathsf{base})$$
$$\alpha : \mathsf{loop} \cdot u \rightsquigarrow u$$

there is a section $\mathsf{R}(u,\alpha) : \prod(t : \mathsf{circle}),\ P(t)$ of P for which there are paths $\mathsf{B}(u,\alpha) : \mathsf{R}(u,\alpha,\mathsf{base}) \rightsquigarrow u$ and a path $\mathsf{L}(u,\alpha)$ witnessing the commutativity of the square

$$\begin{array}{ccc} R(u,\alpha,\mathsf{base}) & \overset{B(u,\alpha)}{\rightsquigarrow} & u \\ {\scriptstyle R(u,\alpha)(\mathsf{loop})}\uparrow & & \uparrow{\scriptstyle \alpha} \\ \mathsf{loop}\cdot R(u,\alpha,\mathsf{base}) & \underset{(\mathsf{loop}\cdot B(u,\alpha))^{-1}}{\leftsquigarrow} & \mathsf{loop}\cdot u \end{array}$$

♦

Lemma 3.2.10. *Suppose A is a type and for $x : A$ there is a path $p : x \rightsquigarrow x$. Then there is a function $f : \mathsf{circle} \to A$ for which we have a path $\mathsf{B} : f(\mathsf{base}) \rightsquigarrow x$ and a commutative square*

$$\begin{array}{ccc} f(x) & \overset{\mathsf{B}}{\rightsquigarrow} & x \\ {\scriptstyle f(\mathsf{loop})}\uparrow & & \uparrow{\scriptstyle p} \\ f(x) & \underset{\mathsf{B}^{-1}}{\leftsquigarrow} & x \end{array}$$

PROOF. We will use the induction principle for the circle to define f. Denote by P the dependent type $\lambda t.A$, take $x : P(\mathsf{base})$. By lemma 3.1.3 there is a path $\alpha : \mathsf{loop} \cdot x \rightsquigarrow x$. So the induction principle gives us a section $f : \mathsf{circle} \to A$ of P for which there are paths $\mathsf{B} : f(\mathsf{base}) \rightsquigarrow x$ and a commutative square

$$\begin{array}{ccc} f(x) & \overset{\mathsf{B}}{\rightsquigarrow} & x \\ {\scriptstyle f(\mathsf{loop})}\uparrow & & \uparrow{\scriptstyle \alpha} \\ \mathsf{loop}\cdot f(x) & \underset{(\mathsf{loop}\cdot \mathsf{B})^{-1}}{\leftsquigarrow} & \mathsf{loop}\cdot x \end{array}$$

The commutativity of the asserted square now follows from lemma 3.1.3. ∎

Theorem 3.2.11 (Correspondence theorem). *For any dependent type P over the circle, the space $\prod(t:\mathsf{circle}),\ P(t)$ of sections of P is equivalent to the space*

$$\mathsf{Constr}(P) :\equiv \sum(u:P(\mathsf{base})),\ \mathsf{loop}\cdot u \rightsquigarrow u$$

of initial data for the induction principle for the circle.

PROOF. For each section $f:\prod(t:\mathsf{circle}),\ P(t)$ of P we get the pair

$$\varphi(f) :\equiv \langle f(\mathsf{base}), f(\mathsf{loop})\rangle$$

in $\mathsf{Constr}(P)$. The induction principle gives a section $\psi(u,p)$ for each pair $\langle u,p\rangle$ in $\mathsf{Constr}(P)$. Thus, we need to show that $\psi\circ\varphi \sim \mathrm{idmap}$ and that $\varphi\circ\psi \sim \mathrm{idmap}$.

Denote $\psi(\varphi(f))$ by $\tilde{f}$, we will verify that there is a homotopy from $\tilde{f}$ to f, i.e.

$$\prod(t:\mathsf{circle}),\ \tilde{f}(t) \rightsquigarrow f(t)$$

We prove this with the induction principle for the circle, using the dependent type Q over the circle given by $Q(t) :\equiv \tilde{f}(t) \rightsquigarrow f(t)$. Since $\tilde{f}$ is obtained from the pair $\langle f(\mathsf{base}), f(\mathsf{loop})\rangle$ by induction, there are paths

$$\begin{aligned}&\alpha : \tilde{f}(\mathsf{base}) \rightsquigarrow f(\mathsf{base})\\ &\beta : \alpha \bullet \tilde{f}(\mathsf{loop}) \bullet P(\mathsf{loop})(\alpha)^{-1} \rightsquigarrow f(\mathsf{loop})\end{aligned}$$

Note that lemma 3.1.1 gives a path

$$Q(\mathsf{loop})(\alpha) \rightsquigarrow f(\mathsf{loop}) \bullet P(\mathsf{loop})(\alpha) \bullet \tilde{f}(\mathsf{loop})^{-1},$$

so a path from $\mathsf{loop}\cdot\alpha = Q(\mathsf{loop})(\alpha) \rightsquigarrow \alpha$ is obtained with a slight modification of β. Therefore, the induction principle for the circle gives indeed a homotopy from $\tilde{f}$ to f. By the function extensionality principle, we get a path $\varphi(\psi(f)) \rightsquigarrow f$.

Now suppose that $u : P(\mathsf{base})$ and that $p : \mathsf{loop}\cdot u \rightsquigarrow u$. This gives the section $f :\equiv \psi(u,p)$ of P for which there are paths

$$\alpha : f(\mathsf{base}) \rightsquigarrow u \qquad \text{and} \qquad \beta : \alpha \bullet f(\mathsf{loop}) \bullet P(\mathsf{loop})(\alpha)^{-1} \rightsquigarrow p.$$

By lemma 2.2.14, we get a path from $\langle f(\mathsf{base}), f(\mathsf{loop})\rangle$ to $\langle u,p\rangle$ from the path α and a path from $\alpha\cdot f(\mathsf{loop})$ to p. By lemma 3.1.2 we have $\alpha\cdot f(\mathsf{loop}) \rightsquigarrow \alpha \bullet f(\mathsf{loop}) \bullet P(\mathsf{loop})(\alpha)^{-1}$. From the latter there is a path to p by β. This finishes the proof of the homotopy from $\varphi\circ\psi$ to idmap. ∎

Corollary 3.2.12. *For any type A we have an equivalence*

$$\left(\sum(x:A),\ x \rightsquigarrow x\right) \simeq (\mathsf{circle} \to A).$$

The following result was first proved by Guillaume Brunerie.

Lemma 3.2.13. *Suppose that C is a type which has a term $b : C$ and a path $l : b \rightsquigarrow b$ and that C has the property that for every dependent type P over C with*

$$u : P(b) \qquad \text{and} \qquad \alpha : l \cdot u \rightsquigarrow u$$

there is a section $f : \prod(t : C),\ P(t)$ for which there are paths $B : f(b) \rightsquigarrow u$ and $L : B \bullet f(l) \bullet l \cdot B \rightsquigarrow \alpha$. Then C is equivalent to circle.

PROOF. Since C satisfies the same induction principle as the circle, an immediate adjustment of corollary 3.2.12 gives equivalences

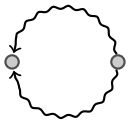
$$\left(\sum(x : A),\ x \rightsquigarrow x\right) \simeq C \to A$$

for each type A. Hence the pair $\langle \mathsf{base}, \mathsf{loop} \rangle$ determines a function $f : C \to \mathsf{circle}$ with the property that there are paths $\beta : f(b) \rightsquigarrow \mathsf{base}$ and $\beta \cdot f(l) \rightsquigarrow \mathsf{loop}$. Also, the pair $\langle b, l \rangle$ determines a function $g : \mathsf{circle} \to C$ with the property that there are paths $\gamma : g(\mathsf{base}) \rightsquigarrow b$ and $\gamma \cdot f(\mathsf{loop}) \rightsquigarrow l$.

To verify that $g \circ f \sim \mathrm{idmap}_C$, note that the homotopy fiber of the mentioned equivalence are contractible. Hence it suffices to verify that $g \circ f$ is in the homotopy fiber of the pair $\langle b, l \rangle$, which corresponds to the identity map on C. For this, it is enough to check that $\langle g(f(b)), g(f(l)) \rangle \rightsquigarrow \langle b, l \rangle$, which follows straight from the stated properties of f and g. The homotopy $f \circ g \sim \mathrm{idmap}_{\mathsf{circle}}$ follows by swapping the roles of C and circle. ∎

3.2.3 An alternative circle

In this subsection we will look at an alternative definition of the circle, which looks like this



The only thing which we show about this circle is that it is equivalent to the original circle. Nevertheless, we will define n-spheres for arbitrary n in section 3.6. The 1-sphere will be this version of the circle.

Definition 3.2.14. We define the type circle' inductively with basic constructors

$$\mathsf{a}, \mathsf{b} : \mathsf{circle}' \qquad \text{and} \qquad \mathsf{p}, \mathsf{q} : \mathsf{a} \rightsquigarrow \mathsf{b}$$

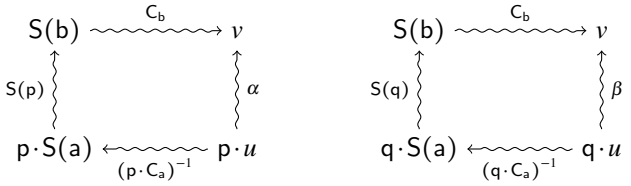
The induction principle for circle' is that whenever P is a dependent type over circle' for which there are

$$u : P(\mathsf{a}) \qquad \alpha : \mathsf{p} \cdot u \rightsquigarrow v$$
$$v : P(\mathsf{b}) \qquad \beta : \mathsf{q} \cdot u \rightsquigarrow v,$$

then there is a section $\mathsf{S} : \prod(x : \mathsf{circle}'),\ P(x)$ for which there are paths

$$\mathsf{C_a} : \mathsf{S(a)} \rightsquigarrow u \qquad \text{and} \qquad \mathsf{C_b} : \mathsf{S(b)} \rightsquigarrow v$$

and paths $\mathsf{K_p}$ and $\mathsf{K_q}$ witnessing the commutativity of the diagrams



Lemma 3.2.15. *The types* circle *and* circle' *are equivalent.*

PROOF. Define $\mathsf{base}' := \mathsf{a}$ and $\mathsf{loop}' := \mathsf{q}^{-1} \bullet \mathsf{p}$. We will show that circle' satisfies the induction principle of circle with this base point and loop.

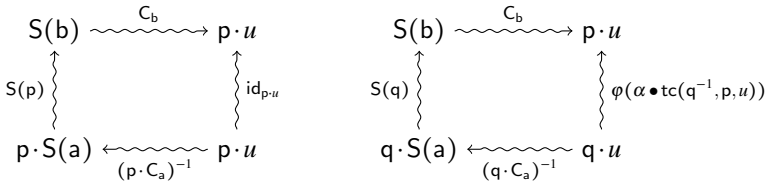
Suppose that P is a dependent type over circle', that $u : P(\mathsf{base}')$ and that $\alpha : \mathsf{loop}' \cdot u \rightsquigarrow u$. To find a section of P, we wish to apply the induction principle for circle', so we must find an element $v : P(\mathsf{b})$ and paths $\mathsf{p} \cdot u \rightsquigarrow v$ and $\mathsf{q} \cdot u \rightsquigarrow v$. We take $v := \mathsf{p} \cdot u$, which gives us the identity path $\mathsf{p} \cdot u \rightsquigarrow v$. Recall from lemma 2.2.5 that there is a function $\mathsf{mvTransp}(q^{-1}) : (u \rightsquigarrow \mathsf{q}^{-1} \cdot v) \to (\mathsf{q} \cdot u \rightsquigarrow v)$. Now note that we have a canonical path

$$\mathsf{q}^{-1} \cdot v = \mathsf{q}^{-1} \cdot (\mathsf{p} \cdot u) \rightsquigarrow (\mathsf{q}^{-1} \bullet \mathsf{p}) \cdot u$$

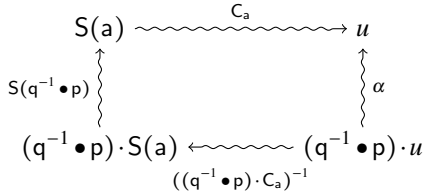
given by $\mathsf{tc}(\mathsf{q}^{-1}, \mathsf{p}, u)$. From the latter there is a path to u by assumption. This completes the induction argument, and hence there is a section S of P for which there are paths

$$\mathsf{C_a} : \mathsf{S(a)} \rightsquigarrow u \qquad \text{and} \qquad \mathsf{C_b} : \mathsf{S(b)} \rightsquigarrow \mathsf{p} \cdot u$$

and paths $\mathsf{K_p}$ and $\mathsf{K_q}$ witnessing the commutativity of the squares



Since we have $\mathsf{C_a} : \mathsf{S}(\mathsf{base}') \rightsquigarrow u$, it is left to check that the diagram



commutes. Recall from lemma 2.2.4 that there is a path

$$\mathsf{S}(\mathsf{q}^{-1} \bullet \mathsf{p}) \rightsquigarrow \mathsf{S}(\mathsf{q}^{-1}) \bullet \mathsf{S}(\mathsf{p}) \bullet \mathsf{tc}(\mathsf{q}^{-1}, \mathsf{p}, \mathsf{S}(\mathsf{a}))^{-1}$$ ∎

3.3 The fundamental groupoid of the circle

In this section we present the result of Mike Shulman that the univalence axiom implies that the fundamental groupoid of the circle is the integers. The approach here to the notion of fundamental groupoid is quite ad hoc, we will actually show that the space $\mathsf{base} \rightsquigarrow \mathsf{base}$ is equivalent to the type $\mathbb{Z}$ of integers.

3.3.1 The type of integers

Definition 3.3.1. We define the type $\mathbb{Z}$ as an inductive space with basic constructors

$$\begin{aligned} 0 &: \mathbb{Z} \\ \mathsf{pos} &: \mathbb{N} \to \mathbb{Z} \\ \mathsf{neg} &: \mathbb{N} \to \mathbb{Z}. \end{aligned}$$

The induction principle for the integers is that whenever P is a dependent type over $\mathbb{Z}$ for which there are

$$\begin{aligned} u &: P(0) \\ k &: \textstyle\prod(n : \mathbb{N}),\ P(\mathsf{pos}(n)) \\ l &: \textstyle\prod(n : \mathbb{N}),\ P(\mathsf{neg}(n)) \end{aligned}$$

then there is a section f of P such that $f(0) = u$, $f(\mathsf{pos}(n)) = k(n)$ and $f(\mathsf{neg}(n)) = l(n)$. ⧫

Note that in this definition we should think of $\mathsf{pos}(0)$ as 1 and of $\mathsf{neg}(0)$ as -1. This is just a minor complication compared to those which arise from the definition of pairs of natural numbers identified appropriately, which the person with the set theoretical approach in mind might be inclined to give.

Lemma 3.3.2. *There is an equivalence* $\mathsf{succ} : \mathbb{Z} \simeq \mathbb{Z}$ *mapping each integer to its successor.*

PROOF. We will use the induction principle for $\mathbb{Z}$. Define $\mathsf{succ}(0) :\equiv \mathsf{pos}(0)$ and define $\mathsf{succ}(\mathsf{pos}(n)) :\equiv \mathsf{pos}(\mathsf{S}(n))$. For the negative integers we use induction on $\mathbb{N}$: define $\mathsf{succ}(\mathsf{neg}(0)) :\equiv 0$ and $\mathsf{succ}(\mathsf{neg}(\mathsf{S}(n))) :\equiv \mathsf{neg}(n)$.

The inverse of succ is the function $\mathsf{pred} : \mathbb{Z} \to \mathbb{Z}$, which is defined in the same way as succ but with the roles of pos and neg interchanged. One might readily check that this gives a homotopy inverse of succ. ∎

Another fact about $\mathbb{Z}$ is that it is a set. We state this property but omit the proof.

Theorem 3.3.3. *The type* $\mathbb{Z}$ *of natural numbers forms a set: for any* $p, q : a \rightsquigarrow b$ *there is a path* $s : p \rightsquigarrow q$.

3.3.2 The universal covering of the circle

In the following lemma we construct a function $\mathbb{Z} \to (\mathsf{base} \rightsquigarrow \mathsf{base})$ which maps an integer a to the loop loop^a.

Lemma 3.3.4. *There is a function* $\mathsf{wind} : \mathbb{Z} \to (\mathsf{base} \rightsquigarrow \mathsf{base})$ *mapping* $\mathsf{pos}(0)$ *to* loop.

PROOF. Define $\mathsf{wind}(0) :\equiv \mathsf{id}_{\mathsf{base}}$, define $\mathsf{wind}(\mathsf{pos}(0)) :\equiv \mathsf{loop}$ and define

$$\mathsf{wind}(\mathsf{pos}(\mathsf{S}(n))) :\equiv \mathsf{wind}(\mathsf{pos}(n)) \bullet \mathsf{loop}.$$

On the negative integers define $\mathsf{wind}(\mathsf{neg}(0)) :\equiv \mathsf{loop}^{-1}$ and

$$\mathsf{wind}(\mathsf{neg}(\mathsf{S}(n))) :\equiv \mathsf{wind}(\mathsf{neg}(n)) \bullet \mathsf{loop}^{-1}. \qquad \blacksquare$$

To show that $(\mathsf{base} \rightsquigarrow \mathsf{base}) \simeq \mathbb{Z}$, we first need the universal cover of the circle. The universal cover is a dependent type over the circle, which we will define using the induction principle for the circle. Note that to give a dependent type over the circle, it suffices to give a type X above the base point and a path from X to X. This is where the univalence axiom comes in: to give a path from X to X it suffices to give an equivalence from X to X.

Definition 3.3.5. The universal cover $U : \mathsf{circle} \to \mathsf{Type}$ is the dependent type defined by $U(\mathsf{base}) :\equiv \mathbb{Z}$ and the loop $\upsilon(\mathbb{Z},\mathbb{Z})^{-1}(\mathsf{succ}) : \mathbb{Z} \rightsquigarrow \mathbb{Z}$. ♦

Lemma 3.3.6. *For any integer a, there is a path* $\mathsf{wind}(a) \cdot a \rightsquigarrow 0$*, where the transport is taken with respect to* U.

PROOF. Since $\mathsf{wind}(0) = \mathsf{id}_{\mathsf{base}}$ it is immediate that there is a path $\mathsf{wind}(0) \cdot 0 \rightsquigarrow 0$. Since $\mathsf{wind}(\mathsf{pos}(0)) = \mathsf{loop}$ ■

Theorem 3.3.7. *The total space* $\sum(t : \mathsf{circle}),\ U(t)$ *of the universal covering of the circle is contractible.*

PROOF. As the center of contraction, we take the pair $\langle \mathsf{base}, 0 \rangle$. To prove the asserted contractibility, it therefore suffices to show that there is a function φ of type

$$\prod(t : \mathsf{circle})(a : U(t)),\ \langle t, a \rangle \rightsquigarrow \langle \mathsf{base}, 0 \rangle,$$

which we find using the induction principle of the circle. First, we need to find a function of type

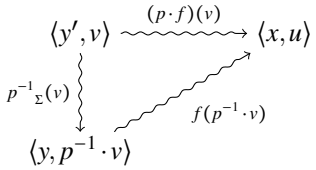
$$\prod(a : \mathbb{Z}),\ \langle \mathsf{base}, a \rangle \rightsquigarrow \langle \mathsf{base}, 0 \rangle.$$

For $a : \mathbb{Z}$, such a path is given by $\mathsf{wind}(a)^{-1}$ as the base path and the path from lemma 3.3.6 as the fiber path. Thus we get the desired function φ. It is left to find a path from $\mathsf{loop} \cdot \varphi$ to φ. The following lemma tells us how to evaluate the function $\mathsf{loop} \cdot \varphi$:

Lemma 3.3.8. *Suppose that P is a dependent type over a type A. For any $x : A$ and $u : P(x)$ define the dependent type $Q : A \to \mathsf{Type}$ by*

$$Q(y) :\equiv \prod(v : P(y)),\ \langle y, v\rangle \rightsquigarrow \langle x, u\rangle.$$

Then the triangle



commutes for every path $p : y \rightsquigarrow y'$ in A, every $f : Q(y)$ and every $v : P(y')$. Recall lemma 2.1.8 for the definition of $p_{\Sigma}(u)$.

PROOF. Immediate by induction on p. ∎

Thus, we see that there is a path

$$(\mathsf{loop} \cdot \varphi)(a) \rightsquigarrow \varphi(\mathsf{loop}^{-1} \cdot a) \bullet {\mathsf{loop}^{-1}}_{\Sigma}(a)$$

for every $a : \mathbb{Z}$. Note that the base path of $\varphi(\mathsf{loop}^{-1} \cdot a) \bullet \mathsf{loop}_{\Sigma}^{-1}(a)$ is identical to $\mathsf{wind}(\mathsf{pred}(a))^{-1} \bullet \mathsf{loop}^{-1}$. Induction on a reveals that the latter path is identical to $\mathsf{wind}(a)^{-1}$. The fiber paths are identical since $\mathbb{Z}$ is a set. This finishes the proof that $\sum(t : \mathsf{circle}),\ U(t)$ is contractible. ∎

Theorem 3.3.9. *There is a transformation $\tau : \prod(t : \mathsf{circle}),\ U(t) \to \mathcal{Y}(\mathsf{base})$. Since both $\sum(t : \mathsf{circle}),\ U(t)$ and $\sum(t : \mathsf{circle}),\ \mathcal{Y}(\mathsf{base})(t)$ are contractible, it follows immediately that*

$$\mathbb{Z} = U(\mathsf{base}) \simeq \mathcal{Y}(\mathsf{base})(\mathsf{base}) = \mathsf{base} \rightsquigarrow \mathsf{base}.$$

PROOF. We already have the function $\mathsf{wind} : \mathbb{Z} \to (\mathsf{base} \rightsquigarrow \mathsf{base})$. Therefore, it is only left to show that $\mathsf{loop} \cdot \mathsf{wind} \rightsquigarrow \mathsf{wind}$. A general argument gives that

$$\mathsf{loop} \cdot \mathsf{wind} \rightsquigarrow \mathcal{Y}(\mathsf{base})(\mathsf{loop}) \circ \mathsf{wind} \circ U(\mathsf{loop}).$$

Thus, we have to show that there is a path

$$\mathsf{wind}(\mathsf{succ}(a)) \bullet \mathsf{loop}^{-1} \rightsquigarrow \mathsf{wind}(a)$$

for any integer a. This is immediate with induction on a. ∎

3.4 Basic properties of paths between paths

The main aim of this section is to develop those properties which are needed to give an inductive definition of a space which has paths of paths, i.e. 2-cells, among the basic constructors. We want to know how homotopies of sections of dependent types behave with respect to paths of paths. In the case of paths, we saw that the essential property was that the diagram

$$\begin{array}{ccc} f(y) & \overset{\alpha(y)}{\rightsquigarrow} & g(y) \\ {\scriptstyle f(p)}\uparrow & & \uparrow{\scriptstyle g(p)} \\ p\cdot f(x) & \underset{p\cdot\alpha(x)}{\rightsquigarrow} & p\cdot g(x), \end{array}$$

commutes for any $\alpha : f \rightsquigarrow g$. The analogous commuting diagram, which we will derive in (5), is

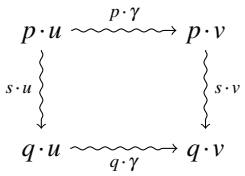
$$\begin{array}{ccc} (\alpha\cdot f^{\rightsquigarrow})(q) & \overset{\tilde{\alpha}(q)}{\rightsquigarrow} & g^{\rightsquigarrow}(q) \\ {\scriptstyle (\alpha\cdot f^{\rightsquigarrow})^{\rightsquigarrow}(s)}\uparrow & & \uparrow{\scriptstyle g^{\rightsquigarrow\rightsquigarrow}(s)} \\ s\cdot((\alpha\cdot f^{\rightsquigarrow})(p)) & \underset{s\cdot\tilde{\alpha}(p)}{\rightsquigarrow} & s\cdot g^{\rightsquigarrow}(p). \end{array}$$

However, the situation complicates at this point, since in this diagram α appears in it's bounded form, e.g. in the transportation $\alpha\cdot f^{\rightsquigarrow}$. The transportation is there because $f^{\rightsquigarrow}(q) : q\cdot f(x) \rightsquigarrow f(y)$ is not in the same space as $g^{\rightsquigarrow}(q) : q\cdot g(x) \rightsquigarrow g(y)$. In its tied form, α is not applicable in the inductive definition of a space because there we need to express how a section induced by some data agrees with that data. These data do not come from a section, so we need to say separately for every component of the data how the induced section agrees with it. We will see that we can replace diagram (5) with two diagrams that are usable in the inductive definition of spaces with paths between paths.

In this section, we assume that $P : A \to \mathsf{Type}$ is a dependent type over A, that $x, y : A$, $p, q : x \rightsquigarrow y$ and that $s : p \rightsquigarrow q$. First of all, note that s immediately induces a path $s^{[-1]} : p^{-1} \rightsquigarrow q^{-1}$.

Lemma 3.4.1. *Write $P(p)$ and $P(q)$ for the transportation functions from $P(x)$ to $P(y)$ induced by p, resp. q. Then there is a homotopy $P(s) : P(p) \rightsquigarrow P(q)$. We denote $P(s)(u) : p\cdot u \rightsquigarrow q\cdot u$ by $s\cdot u$.*

If $\gamma : u \rightsquigarrow v$ is a path in $P(x)$, then the square



commutes by a path we denote by $\mathsf{s} \cdot \gamma$.

PROOF. Immediate with induction on s. ∎

Suppose furthermore that $f : \prod(x : A),\ P(x)$ is a section of P. Write $f^{\rightsquigarrow}$ for the action on paths of A that f induces, i.e.

$$f^{\rightsquigarrow} : \prod(x, y : A)(p : x \rightsquigarrow y),\ p \cdot f(x) \rightsquigarrow f(y).$$

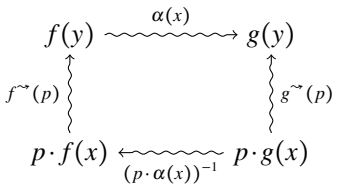
Recall that this function is defined in the dependent map lemma 2.1.7. Then we get, for each $x, y : A$ the dependent function $f^{\rightsquigarrow}(x, y) : \prod(p : x \rightsquigarrow y),\ p \cdot f(x) \rightsquigarrow f(y)$ which is a section of the dependent type $P^{\rightsquigarrow}$ over $x \rightsquigarrow y$ given by $P^{\rightsquigarrow}(p) :\equiv p \cdot f(x) \rightsquigarrow f(y)$. Hence we have

$$(f^{\rightsquigarrow}(x, y))^{\rightsquigarrow} : \prod(p, q : x \rightsquigarrow y)(s : p \rightsquigarrow q),\ s \cdot f^{\rightsquigarrow}(x, y, p) \rightsquigarrow f^{\rightsquigarrow}(x, y, q).$$

It is useful to keep the notation $f^{\rightsquigarrow}$, but we will omit reference to x and y and we will simply write $f^{\rightsquigarrow}(p)$ instead of $f^{\rightsquigarrow}(x, y, p)$.

If we suppose that $g : \prod(x : A),\ P(x)$ is also a section of P, for which there is a path $\alpha : f \rightsquigarrow g$, then we get paths $\alpha(x) : f(x) \rightsquigarrow g(x)$ for each $x : A$. We also get the path $P(p)^{\rightsquigarrow}(\alpha(x)) : p \cdot f(x) \rightsquigarrow p \cdot f(y)$, which we denote simply by $p \cdot \alpha(x)$. The following lemma says that we also get paths $\alpha(p)$.

Lemma 3.4.2. *Suppose that $\alpha : f \rightsquigarrow g$ is a path in $\prod(x : A),\ P(x)$ and that $p : x \rightsquigarrow y$ is a path in A. Then there is a path $\alpha(p)$ witnessing the commutativity of the square*



PROOF. This is immediate with induction on α and p. ∎

Lemma 3.4.3. *Suppose that $p, q : x \rightsquigarrow y$ are paths in A and that $s : p \rightsquigarrow q$. Then, for any dependent type P over A there is a term of type*

$$\prod(u : P(x))(v : P(y))(\gamma : p \cdot u \rightsquigarrow v),\ \gamma \bullet (s \cdot u)^{-1} \rightsquigarrow s \cdot \gamma.$$

As an immediate consequence, we have for any section $f : \prod(x:A),\ P(x)$ of P that there is a path $f(s)' : f(p) \bullet (s \cdot u)^{-1} \rightsquigarrow f(q)$.

PROOF. With induction over s. The lemma is obvious when $s = \mathsf{id}_p : p \rightsquigarrow p$. The path $f(s)'$ is then given by the composition of $f(s)$ with the path of this lemma at the triple $(f(x), f(y), f(p))$. ∎

Lemma 3.4.4. *Suppose that $\alpha : f \rightsquigarrow g$ is a path in the space $\prod(x:A),\ P(x)$. Then define the dependent space Q over $\prod(x:A),\ P(a)$ by*

$$Q(f) :\equiv \prod(x,y:A)(p:x \rightsquigarrow y),\ p \cdot f(x) \rightsquigarrow f(y).$$

It follows immediately that $f^{\rightsquigarrow} : Q(f)$ and that there is a path $\tilde{\alpha} : \alpha \cdot f^{\rightsquigarrow} \rightsquigarrow g^{\rightsquigarrow}$ in $Q(g)$. The values of $\alpha \cdot f^{\rightsquigarrow}$ are also computed by

$$(\alpha \cdot f^{\rightsquigarrow})(p) \rightsquigarrow \alpha(y) \bullet f^{\rightsquigarrow}(p) \bullet (p \cdot \alpha(x))^{-1}$$

for $x, y : A$ and $p : x \rightsquigarrow y$. If we have $p, q : x \rightsquigarrow y$ and $s : p \rightsquigarrow q$, then the diagram

$$\begin{array}{ccc} (\alpha \cdot f^{\rightsquigarrow})(q) & \xrightarrow{\tilde{\alpha}(q)} & g^{\rightsquigarrow}(q) \\ {\scriptstyle (\alpha \cdot f^{\rightsquigarrow})^{\rightsquigarrow}(s)}\uparrow & & \uparrow{\scriptstyle g^{\rightsquigarrow\rightsquigarrow}(s)} \\ s \cdot (\alpha \cdot f^{\rightsquigarrow})(p) & \xrightarrow[s \cdot (\tilde{\alpha}(p))]{} & s \cdot g^{\rightsquigarrow}(p) \end{array} \tag{5}$$

commutes. All the terms and paths in this diagram are in the space $q \cdot g(x) \rightsquigarrow g(y)$. We can express those terms as a composition of already known paths, using induction on α. We have pictured these in figure **??**.

PROOF. All the assertions made in this lemma are proved with path induction. ∎

Lemma 3.4.5. *Suppose that $p, q : x \rightsquigarrow y$ are paths in A and that $s : p \rightsquigarrow q$. Suppose furthermore that $P : A \to \mathsf{Type}$ is a dependent type over A and that there are paths*

$$\alpha : u \rightsquigarrow u' \quad \text{in } P(x), \qquad \beta : v \rightsquigarrow v' \quad \text{in } P(y), \qquad \gamma : p \cdot u \rightsquigarrow v \quad \text{in } P(y).$$

Then there is a path witnessing the commutativity of the square

$$\begin{array}{ccc} v & \xrightarrow{\beta} & v' \\ {\scriptstyle s \cdot \gamma}\uparrow & & \uparrow{\scriptstyle s \cdot (\beta \bullet \gamma \bullet (p \cdot \alpha)^{-1})} \\ q \cdot u & \xleftarrow[q \cdot \alpha]{} & q \cdot u' \end{array}$$

of paths in $P(y)$. Here, the transportation $s \cdot \gamma$ is taken with respect to the dependent type D over $x \rightsquigarrow y$ given by $D(p) :\equiv p \cdot u \rightsquigarrow v$ and the transportation

$$s \cdot (\beta \bullet \gamma \bullet (p \cdot \alpha)^{-1})$$

is taken with respect to the dependent type D' over $x \rightsquigarrow y$ given by $D'(p) :\equiv p \cdot u' \rightsquigarrow v'$.

PROOF. Immediate with induction on s. ∎

Lemma 3.4.6 (Untying lemma). *Suppose that P is a dependent type over A and that $\alpha : f \rightsquigarrow g$ is a path in the space $\prod(x:A),\ P(x)$ of sections of P the space of paths witnessing the commutativity of the square*

$$\begin{array}{ccc} (\alpha \cdot f^{\rightsquigarrow})(q) & \xrightarrow{\tilde{\alpha}(q)} & g^{\rightsquigarrow}(q) \\ {\scriptstyle (\alpha \cdot f^{\rightsquigarrow})^{\rightsquigarrow}(s)}\uparrow & & \uparrow{\scriptstyle g^{\rightsquigarrow\rightsquigarrow}(s)} \\ s \cdot (\alpha \cdot f^{\rightsquigarrow})(p) & \xrightarrow[s \cdot (\tilde{\alpha}(p))]{} & s \cdot g^{\rightsquigarrow}(p) \end{array}$$

is equivalent to the space of paths witnessing the commutativity of the pentagon

$$\begin{array}{ccccc} \alpha(y) \bullet (s \cdot f(p)) \bullet (q \cdot \alpha(x))^{-1} & & \xrightarrow{\alpha(y) \bullet f(s) \bullet (q \cdot \alpha(x))^{-1}} & & \alpha(y) \bullet f(q) \bullet (q \cdot \alpha(x))^{-1} \\ \downarrow & & & & \downarrow{\scriptstyle \alpha(q)} \\ s \cdot (\alpha(y) \bullet f(p) \bullet (p \cdot \alpha(x))^{-1}) & \xrightarrow[s \cdot \alpha(p)]{} & s \cdot g(p) & \xrightarrow[g(s)]{} & g(q) \end{array}$$

The unlabeled path in this diagram is the canonical path from lemma 3.4.5.

PROOF. Using induction on α, s and p, both of the spaces reduce to $\mathsf{id}_{f(x)} \rightsquigarrow \mathsf{id}_{f(x)}$. ∎

3.5 Some examples of inductive spaces with 2-cells

We are now ready to give our initial couple of examples of inductive spaces which have paths between paths among their basic constructors. We will give two of them, the disc and the sphere. The disc plays the role of the interval, in the sense that it is the minimal example in which every path is nontrivial, meaning that it has two different points, two different paths between them and a path between those paths. The sphere has, like the circle, only one base point and it has a path from the identity path on that base point to itself which represents the surface of the sphere.

3.5.1 The disc

Definition 3.5.1. Define the disc inductively as a type disc with basic constructors $\mathsf{a}, \mathsf{b} : \mathsf{disc}$, $\mathsf{p}, \mathsf{q} : \mathsf{a} \rightsquigarrow \mathsf{b}$ and $\mathsf{d} : \mathsf{p} \rightsquigarrow \mathsf{q}$. The induction principle for the disc is that whenever we

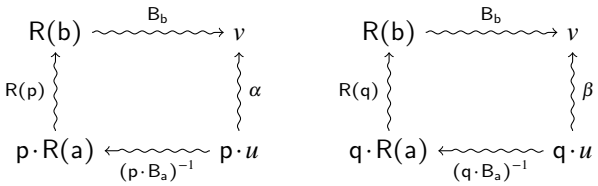
have

$$\begin{aligned} &P : \mathsf{disc} \to \mathsf{Type} && \alpha : \mathsf{p} \cdot u \rightsquigarrow v \\ &u : P(\mathsf{a}) && \beta : \mathsf{q} \cdot u \rightsquigarrow v \\ &v : P(\mathsf{b}) && \gamma : s \cdot \alpha \rightsquigarrow \beta \end{aligned}$$

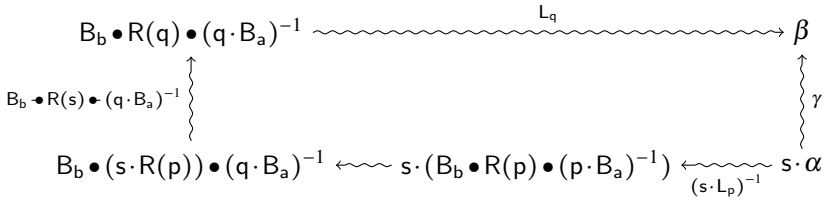
then there is a section $\mathsf{R} : \prod(x : \mathsf{disc}),\ P(x)$ for which there are paths

$$\mathsf{B_a} : \mathsf{R}(\mathsf{a}) \rightsquigarrow u, \quad \text{and} \quad \mathsf{B_b} : \mathsf{R}(\mathsf{b}) \rightsquigarrow v,$$

there are paths $\mathsf{L_p}$ and $\mathsf{L_q}$ witnessing the commutativity of the following diagrams (respectively)



and a path Q witnessing the commutativity of the pentagram



(6)

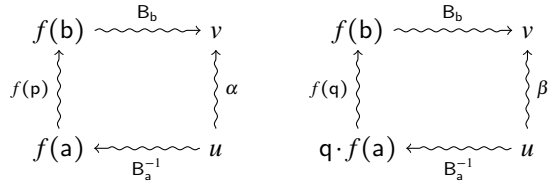
Here, the unlabeled path is the canonical path from lemma 3.4.5. ⧫

We state the non-dependent induction principle for the disc without proof.

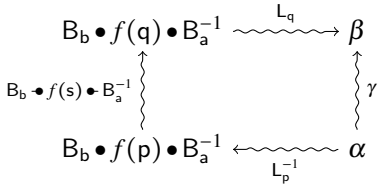
Lemma 3.5.2. *For any paths $\alpha, \beta : x \rightsquigarrow y$ and $\delta : \alpha \rightsquigarrow \beta$ in a type A there is a function*

$$f : \mathsf{disc} \to A$$

for which there are paths $\mathsf{B_a} : f(\mathsf{a}) \rightsquigarrow x$ and $\mathsf{B_b} : f(\mathsf{b}) \rightsquigarrow y$, paths $\mathsf{L_p}$ and $\mathsf{L_q}$ witnessing the commutativity of the squares



and a path Q *witnessing the commutativity of the diagram*



Theorem 3.5.3 (Correspondence theorem). *For any dependent space P over the disc, the space of sections of P is equivalent to the space*

$$\mathsf{Constr}(P) :\equiv \sum(u : P(\mathsf{a}))(v : P(\mathsf{b}))(\alpha : \mathsf{p} \cdot u \rightsquigarrow v)(\beta : \mathsf{q} \cdot u \rightsquigarrow v),\ \mathsf{s} \cdot \alpha \rightsquigarrow \beta.$$

of initial data for sections of P.

PROOF. Suppose that $f : \prod(w : \mathsf{disc}),\ P(w)$ is a section of a dependent space P over the disc. Then we have

$$\begin{aligned} &f(\mathsf{a}) : P(\mathsf{a}) && f(\mathsf{p}) : \mathsf{p} \cdot f(\mathsf{a}) \rightsquigarrow f(\mathsf{b}) \\ &f(\mathsf{b}) : P(\mathsf{b}) && f(\mathsf{q}) : \mathsf{q} \cdot f(\mathsf{a}) \rightsquigarrow f(\mathsf{b}) \\ & && f(\mathsf{s}) : \mathsf{s} \cdot f(\mathsf{p}) \rightsquigarrow f(\mathsf{q}) \end{aligned}$$

which form an element of $\mathsf{Constr}(P)$. This gives us a function φ from $\prod(w : \mathsf{disc}),\ P(w)$ to $\mathsf{Constr}(P)$. The induction principle for the disc gives a section for each element of $\mathsf{Constr}(P)$, so the induction principle is the map ψ from $\mathsf{Constr}(P)$ to $\prod(w : \mathsf{disc}),\ P(w)$.

We will first verify that there is a homotopy from $\psi(\varphi(f))$ to f for every section f of P; the function $\psi(\varphi(f))$ will be denoted by $\tilde{f}$. Thus, we will have to show that

$$\prod(w : \mathsf{disc}),\ \tilde{f}(w) \rightsquigarrow f(w)$$

We will use the induction principle for the disc. Note that there are paths

$$\begin{aligned} u &:\equiv \mathsf{B}_{f(\mathsf{a})} : \tilde{f}(\mathsf{a}) \rightsquigarrow f(\mathsf{a}) \\ v &:\equiv \mathsf{B}_{f(\mathsf{b})} : \tilde{f}(\mathsf{b}) \rightsquigarrow f(\mathsf{b}). \end{aligned}$$

We use a separate lemma to find paths $p \cdot u \rightsquigarrow v$ and $q \cdot u \rightsquigarrow v$.

Lemma 3.5.4. *Suppose that $p : x \rightsquigarrow y$ is a path of type A. For any dependent type $P : A \to \mathsf{Type}$ over A, for any two sections $f, g : \prod(x : A),\ P(x)$ of P and any two paths $\alpha : f(x) \rightsquigarrow g(x)$ and $\beta : f(y) \rightsquigarrow g(y)$, the space*

$$p \cdot \alpha \rightsquigarrow \beta,$$

where the transportation is taken with respect to the dependent type Q over A given by $Q(x) :\equiv f(x) \rightsquigarrow g(x)$, is equivalent to the space of paths witnessing the commutativity of the square

$$\begin{array}{ccc} f(y) & \xrightarrow{\beta} & g(y) \\ {\scriptstyle f(p)}\uparrow & & \uparrow{\scriptstyle g(p)} \\ p \cdot f(x) & \xleftarrow[P(p)^{\rightsquigarrow}(\alpha)^{-1}]{} & p \cdot g(x). \end{array}$$

PROOF. With induction on p the first space reduces to $\alpha \rightsquigarrow \beta$ while the second space reduces to $\beta \bullet \alpha^{-1} \rightsquigarrow \mathsf{id}_{g(x)}$. Those spaces are equivalent. ∎

We apply this equivalence on the paths $\mathsf{L_p}$ and $\mathsf{L_q}$ to get paths $\alpha : p \cdot u \rightsquigarrow v$ and $\beta : q \cdot u \rightsquigarrow v$ respectively. The last step in the induction proof is to find a path from $\mathsf{s} \cdot \alpha$ to β. Again, we formulate a separate lemma with a more general approach.

Lemma 3.5.5. *Suppose that $p, q : x \rightsquigarrow y$ are paths in A and that $s : p \rightsquigarrow q$. Suppose also that $P : A \to \mathsf{Type}$ is a dependent type over A, that $f, g : \prod(x : A),\ P(x)$ are sections of P and that there are paths*

$$\begin{aligned} &\alpha : f(x) \rightsquigarrow g(x) && \gamma : \beta \bullet f(p) \bullet P(p)^{\rightsquigarrow}(\alpha)^{-1} \rightsquigarrow g(p) \\ &\beta : f(y) \rightsquigarrow g(y) && \delta : \beta \bullet f(q) \bullet P(q)^{\rightsquigarrow}(\alpha)^{-1} \rightsquigarrow g(q). \end{aligned}$$

Then the space

$$s \cdot \gamma \rightsquigarrow \delta,$$

where the transportation is taken with respect to the dependent type Q over $x \rightsquigarrow y$ given by $Q(p) :\equiv \beta \bullet f(p) \bullet P(p)^{\rightsquigarrow}(\alpha) \rightsquigarrow g(p)$, is equivalent to the space of paths witnessing the commutativity of the diagram

$$\begin{array}{ccccc} \beta \bullet (s \cdot f(p)) \bullet P(q)^{\rightsquigarrow}(\alpha)^{-1} & & \xrightarrow{\beta \bullet f(s) \bullet P(q)^{\rightsquigarrow}(\alpha)^{-1}} & & \beta \bullet f(q) \bullet P(q)^{\rightsquigarrow}(\alpha)^{-1} \\ \downarrow & & & & \downarrow{\scriptstyle \delta} \\ s \cdot (\beta \bullet f(p) \bullet P(p)^{\rightsquigarrow}(\alpha)^{-1}) & \xrightarrow[s \cdot \gamma]{} & s \cdot g(p) & \xrightarrow[g(s)]{} & g(q) \end{array}$$

PROOF. With induction on s both spaces reduce to $\gamma \rightsquigarrow \delta$. ∎

Note that, by the equivalence of lemma 3.5.4, the space $\mathsf{s} \cdot \alpha \rightsquigarrow \beta$ is equivalent to the space $\mathsf{s} \cdot \mathsf{L_p} \rightsquigarrow \mathsf{L_q}$. Thus, we get the desired path $\mathsf{s} \cdot \alpha \rightsquigarrow \beta$ by applying the above lemma to Q. This concludes our proof with induction that $\tilde{f}$ and f are homotopic.

Now suppose we have a quintuple $\mathbf{x} :\equiv \langle u, v, \alpha, \beta, \gamma \rangle$ of $\mathsf{Constr}(P)$. The last task in order to finish the proof of the correspondence theorem for the disc is to find a path from $\varphi(\psi(\mathbf{x})) \rightsquigarrow \mathbf{x}$.

Note that the induction principle gives the paths $\mathsf{B_a} : \mathsf{R(a)} \rightsquigarrow u$ and $\mathsf{B_b} : \mathsf{R(b)} \rightsquigarrow v$. A path from $\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(p)}) \rightsquigarrow \alpha$ is given by the canonical path

$$\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(p)}) \rightsquigarrow \mathsf{B_b} \bullet \mathsf{R(p)} \bullet P(\mathsf{p})^{\rightsquigarrow} (\mathsf{B_a})^{-1},$$

followed by $\mathsf{L_p}$. Similarly, we get a path from $\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(q)})$ to β using $\mathsf{L_q}$. For the last step, we have to verify that

$$\mathsf{L_q} \cdot (\mathsf{L_p} \cdot (\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(s)}))) \rightsquigarrow \gamma.$$

Note that

$$\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(s)}) \rightsquigarrow \mathsf{B_b} \bullet \mathsf{R(s)} \bullet P(\mathsf{q})^{\rightsquigarrow} (\mathsf{B_a})^{-1}$$

Then $\mathsf{L_q} \cdot (\mathsf{L_p} \cdot (\mathsf{B_b} \cdot (\mathsf{B_a} \cdot \mathsf{R(s)})))$ is propositionally equal to the composition of the solid arrows in the diagram

$$\begin{array}{ccccc}
\mathsf{B_b} \bullet (\mathsf{s} \cdot \mathsf{R(p)}) \bullet P(\mathsf{q})^{\rightsquigarrow} (\mathsf{B_a})^{-1} & & \xrightarrow{\mathsf{B_b} \bullet \mathsf{R(s)} \bullet P(\mathsf{q})(\mathsf{B_a})^{-1}} & & \mathsf{B_b} \bullet \mathsf{R(q)} \bullet P(\mathsf{q})^{\rightsquigarrow} (\mathsf{B_a})^{-1} \\
\uparrow & & & & \downarrow{\scriptstyle \mathsf{L}_\beta} \\
\mathsf{s} \cdot (\mathsf{B_b} \bullet \mathsf{R(p)} \bullet P(\mathsf{p})^{\rightsquigarrow} (\mathsf{B_a})^{-1}) & \xleftarrow[\mathsf{s} \cdot \mathsf{L}_\alpha]{} & \mathsf{s} \cdot \alpha & \xrightarrow[\gamma]{} & \beta
\end{array}$$

The commutativity of this diagram is clearly equivalent to the commutativity of diagram 6, for which we have Q. This finishes the proof that there is a path $\varphi(\psi(\mathbf{x})) \rightsquigarrow \mathbf{x}$, completing also the proof of the correspondence theorem for the disc. ∎

Corollary 3.5.6. *For any type A, we have an equivalence between* $\mathsf{disc} \to A$ *and the space*

$$\sum (x, y : A)(p, q : x \rightsquigarrow y),\ p \rightsquigarrow q.$$

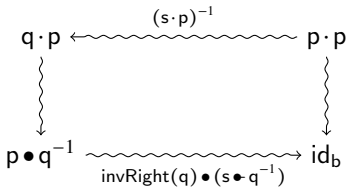
Lemma 3.5.7. *The disc is contractible.*

PROOF. We wish to show with induction that there is a term of type

$$\prod (w : \mathsf{disc}),\ w \rightsquigarrow \mathsf{b}.$$

Let P be the dependent type over disc given by $P(w) := w \rightsquigarrow \mathsf{b}$. We have the terms $\mathsf{p} : P(\mathsf{a})$ and $\mathsf{id_b} : P(\mathsf{b})$. There is a canonical term α of type $\mathsf{p} \cdot \mathsf{p} \rightsquigarrow \mathsf{id}_b$ and a canonical term of type $\mathsf{q} \cdot \mathsf{p} \rightsquigarrow \mathsf{p} \bullet \mathsf{q}^{-1}$. The space $\mathsf{p} \bullet \mathsf{q}^{-1} \rightsquigarrow \mathsf{id_b}$ is equivalent to the space $\mathsf{p} \rightsquigarrow \mathsf{q}$, which is inhabited by the path s. This gives a path $\beta : \mathsf{q} \cdot \mathsf{p} \rightsquigarrow \mathsf{id_b}$. Note that $\mathsf{s} \cdot \alpha \rightsquigarrow \alpha \bullet (\mathsf{s} \cdot \mathsf{p})^{-1}$.

The square



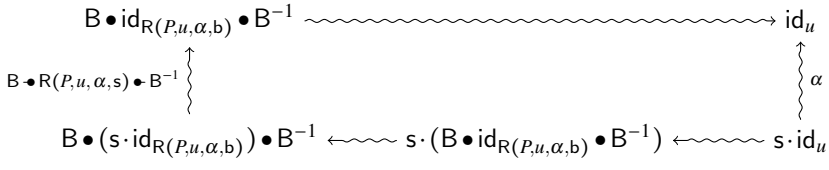
commutes generally (by induction on a general path $s : p \rightsquigarrow q$). ∎

3.5.2 The sphere

Definition 3.5.8. We define the type sphere inductively with basic constructors $\mathsf{b} :$ sphere and $\mathsf{s} : \mathsf{id}_{\mathsf{b}} \rightsquigarrow \mathsf{id}_{\mathsf{b}}$. The induction principle for the sphere is that whenever we have

$$\begin{aligned} &P : \mathsf{sphere} \to \mathsf{Type} \\ &u : P(\mathsf{b}) \\ &\alpha : \mathsf{s} \cdot \mathsf{id}_u \rightsquigarrow \mathsf{id}_u, \end{aligned}$$

where the transportation in $\mathsf{s} \cdot \mathsf{id}_u$ is taken with respect to the dependent type $\lambda p.p \cdot u \rightsquigarrow u$, then there is a section $\mathsf{R}(P,u,\alpha) : \prod(x : \mathsf{sphere}),\ P(x)$ of P for which there is a path $\mathsf{B} := \mathsf{B}(P,u,\alpha) : \mathsf{R}(P,u,\alpha,\mathsf{b}) \rightsquigarrow u$ and a path $\mathsf{Q}(P,u,\alpha)$ witnessing the commutativity of the diagram



♦

Theorem 3.5.9 (Correspondence theorem for sphere**).** *Suppose P is a dependent type over* sphere*. Then there is an equivalence between the space of sections of P and the space*

$$\mathsf{Constr}(P) := \sum(x : P(\mathsf{base}),\ \mathsf{s} \cdot \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$$

of prerequisites for application of the induction principle for the sphere are equivalent.

PROOF. The function $\psi : \mathsf{Constr}(P) \to \prod(t : \mathsf{sphere}),\ P(t)$ is given by the induction principle. The function $\varphi : (\prod(t : \mathsf{sphere},\ P(t)) \to \mathsf{Constr}(P)$ is given by

$$\lambda f.\langle f(\mathsf{base}), f(\mathsf{s})\rangle.$$

We have to show that there are homotopies $\psi \circ \varphi \sim \mathrm{idmap}$ and $\varphi \circ \psi \sim \mathrm{idmap}$.

Suppose $f : \prod(t : \mathsf{sphere}),\ P(t)$ and denote $\psi(\varphi(f))$ by $\tilde{f}$. We will find a homotopy from $\tilde{f}$ to f using the induction principle of the sphere. Note that the induction principle gives us a path $\mathsf{B} : \tilde{f}(\mathsf{base}) \rightsquigarrow f(\mathsf{base})$ and a path Q witnessing the commutativity of the diagram

$$\begin{array}{ccccc}
\mathsf{B} \bullet \mathsf{id}_{\tilde{f}(\mathsf{b})} \bullet \mathsf{B}^{-1} & & \rightsquigarrow & & \mathsf{id}_{f(\mathsf{b})} \\
{\scriptstyle \mathsf{B} \bullet \tilde{f}(\mathsf{s}) \bullet \mathsf{B}^{-1}} \uparrow & & & & \uparrow {\scriptstyle f(\mathsf{s})} \\
\mathsf{B} \bullet (\mathsf{s} \cdot \mathsf{id}_{\tilde{f}(\mathsf{b})}) \bullet \mathsf{B}^{-1} & \leftsquigarrow & \mathsf{s} \cdot (\mathsf{B} \bullet \mathsf{id}_{\tilde{f}(\mathsf{b})} \bullet \mathsf{B}^{-1}) & \leftsquigarrow & \mathsf{s} \cdot \mathsf{id}_{f(\mathsf{b})}
\end{array}$$

By lemma 3.5.5, we can get a path $\mathsf{s} \cdot \mathsf{id}_{\mathsf{B}} \rightsquigarrow \mathsf{id}_{\mathsf{B}}$ from Q. Hence the functions $\tilde{f}$ and f are indeed homotopic and therefore identical.

For the other homotopy, $\varphi \circ \psi \sim \mathrm{idmap}$, we have a separate lemma:

Lemma 3.5.10. *Suppose A is a type, $x : A$ and $s : \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$ and that P is a dependent type over A. Define the dependent type Q over $P(x)$ by*

$$Q(t) :\equiv s \cdot \mathsf{id}_t \rightsquigarrow \mathsf{id}_t.$$

Suppose furthermore that $p : u \rightsquigarrow v$ is a path in $P(x)$ and that $\alpha : Q(u)$ and that $\beta : Q(v)$. Then the space $p \cdot \alpha \rightsquigarrow \beta$ is equivalent to the space of paths witnessing the commutativity of the diagram

$$\begin{array}{ccccc}
p \bullet \mathsf{id}_u \bullet p^{-1} & & \rightsquigarrow & & \mathsf{id}_v \\
{\scriptstyle p \bullet \alpha \bullet p^{-1}} \uparrow & & & & \uparrow {\scriptstyle \beta} \\
p \bullet (s \cdot \mathsf{id}_u) \bullet p^{-1} & \leftsquigarrow & s \cdot (p \bullet \mathsf{id}_u \bullet p^{-1}) & \leftsquigarrow & s \cdot \mathsf{id}_v
\end{array}$$

PROOF. The proof is immediate by induction on p. ∎

This lemma has the immediate consequence that $\varphi(\psi(\langle u, \alpha \rangle)) \rightsquigarrow \langle u, \alpha \rangle$ for each $\langle u, \alpha \rangle$ in $\mathsf{Constr}(P)$. ∎

Corollary 3.5.11. *For any type A, the spaces $\mathsf{sphere} \to A$ and*

$$\sum(x : A),\ \mathsf{id}_x \rightsquigarrow \mathsf{id}_x$$

are equivalent.

Lemma 3.5.12. *The sphere is determined uniquely up to equivalence.*

PROOF. Suppose that S is a type with $b : S$ and $s : \mathsf{id}_b \rightsquigarrow \mathsf{id}_b$ satisfying the same induction principle as the sphere. Then there is a variant of corollary 3.5.11 valid for S. Thus

we obtain a function $\varphi : S \to \mathsf{sphere}$ corresponding to the pair $\langle \mathsf{b}, \mathsf{s} \rangle$ for which there are paths

$$\alpha : \varphi(b) \rightsquigarrow \mathsf{b} \qquad \text{and} \qquad \beta : \alpha \cdot \varphi(s) \rightsquigarrow \mathsf{s}$$

and a function $\psi : \mathsf{sphere} \to S$ corresponding to the pair $\langle b, s \rangle$ for which there are paths

$$\gamma : \psi(\mathsf{b}) \rightsquigarrow b \qquad \text{and} \qquad \delta : \gamma \cdot \psi(\mathsf{s}) \rightsquigarrow s.$$

To see that $\psi \circ \varphi \sim \mathrm{idmap}_S$, it suffices to show that $\psi \circ \varphi$ is in the same homotopy fiber of the equivalence $(\sum(x : S),\ \mathsf{id}_x \rightsquigarrow \mathsf{id}_x) \simeq S \to S$ as the identity map idmap_S. Therefore, it suffices to check that there are paths $\mu : \psi(\varphi(b)) \rightsquigarrow b$ and $\nu : \mu \cdot \psi(\varphi(s)) \rightsquigarrow s$. For the path μ we take $\gamma \bullet \psi(\alpha)$. For the path ν notice that there are paths

$$(\gamma \bullet \psi(\alpha)) \cdot \psi(\varphi(s)) \rightsquigarrow \gamma \cdot (\psi(\alpha) \cdot \psi(\varphi(s))) \rightsquigarrow \gamma \cdot \psi(\alpha \cdot \varphi(s))) \rightsquigarrow \gamma \cdot \psi(\mathsf{s}) \rightsquigarrow s.$$

The proof that there is a homotopy $\varphi \circ \psi \sim \mathrm{idmap}_{\mathsf{sphere}}$ is similar. ∎

3.6 Directed colimits

We take the practical approach and define only the directed limit that we need. However, it can already be used for several interesting constructions.

Definition 3.6.1. Suppose we have a dependent type $A : \mathbb{N} \to \mathsf{Type}$ and suppose that we have a dependent function

$$\alpha : \prod(n : \mathbb{N}),\ A_n \to A_{n+1}.$$

Then we can define the space $A_\omega := [A, \alpha]_\omega$ inductively as the space with basic constructors

$$\begin{aligned} &\beta : \prod(n : \mathbb{N}),\ A_n \to A_\omega \\ &\gamma : \prod(n : \mathbb{N})(x : A_n),\ \beta_{n+1}(\alpha_n(x)) \rightsquigarrow \beta_n(x). \end{aligned}$$

The induction principle for A_ω is that for each dependent type Λ over A_ω for which there are

$$\begin{aligned} &B : \prod(n : \mathbb{N})(x : A_n),\ \Lambda(\beta_n(x)) \\ &\Gamma : \prod(n : \mathbb{N})(x : A_n),\ \gamma_n(x) \cdot B_{n+1}(\alpha_n(x)) \rightsquigarrow B_n(x) \end{aligned}$$

there is a section $s : \prod(w : A_\omega),\ \Lambda(w)$ with the property that there are paths $\mu_n(x) : s(\beta_n(x)) \rightsquigarrow B_n(x)$ and paths $\nu_n(x)$ witnessing the commutativity of the diagram

$$\begin{array}{ccc} s(\beta_n(x)) & \xrightsquigarrow{\mu_n(x)} & B_n(x) \\ {\scriptstyle s(\gamma_n(x))}\uparrow & & \uparrow{\scriptstyle \Gamma_n(x)} \\ \gamma_n(x) \cdot s(\beta_{n+1}(\alpha_n(x))) & \xleftarrow[(\gamma_n(x) \cdot \mu_{n+1}(\alpha_n(x)))^{-1}]{} & \gamma_n(x) \cdot B_{n+1}(\alpha_n(x)) \end{array}$$

for every $n : \mathbb{N}$ and every $x : A_n$. ♦

The following lemma is just a sanity check, to see if the universal property as stated works out nicely in a simple case:

Lemma 3.6.2. *The directed colimit of the sequence*

$$A \xrightarrow{\mathsf{id}_A} A \xrightarrow{\mathsf{id}_A} A \xrightarrow{\mathsf{id}_A} \cdots$$

is equivalent to A.

PROOF. We use the induction principle to find a function from A_ω to A. For each $n : \mathbb{N}$ take B_n to be the identity map idmap_A. Then we have $B_{n+1}(x) = x$ for each $x : A$ and hence we get a function $r : A_\omega \to A$ with the property that there is a homotopy $\mu_n : r \bullet \beta_n \sim \mathrm{idmap}_A$ for each $n : \mathbb{N}$. We wish to show that β_0 is the inverse of r. Note that it is only left to show that there is a section of the dependent type Λ over A_ω given by

$$\Lambda(w) := \beta_0(r(w)) \rightsquigarrow w,$$

which we do with the induction principle for A_ω. Thus, we have to give maps

$$B : \prod(n : \mathbb{N})(x : A_n),\ \beta_0(r(\beta_n(x))) \rightsquigarrow \beta_n(x)$$
$$\Gamma : \prod(n : \mathbb{N})(x : A_n),\ \gamma_n(x) \cdot B_{n+1}(x) \rightsquigarrow B_n(x)$$

We define B by induction: $B_0(x) := \beta_0(\mu_0(x))$ and $B_{n+1}(x) := B_n(x) \bullet \gamma_n(x)$. Since there is a canonical path $\gamma_n(x) \cdot B_{n+1}(x) \rightsquigarrow B_{n+1}(x) \bullet \gamma_n(x)^{-1}$, we immediately get Γ. ∎

Another case where it is easy to compute the directed colimit is where each of the functions α_n is constant. We will use this case to show that the ω-sphere is contractible.

Lemma 3.6.3. *Suppose that $A : \mathbb{N} \to \mathsf{Type}$ is a dependent type over $\mathbb{N}$ and that $a : \prod(n : \mathbb{N}), A_n$ is a section of A. Define $\alpha : \prod(n : \mathbb{N}), A_n \to A_{n+1}$ to be $\lambda n.\lambda x.a_{n+1}$. Then the type A_ω is contractible.*

PROOF. We will show, using the induction principle for A_ω, that there is a term of type

$$\prod(x : A_\omega),\ x \rightsquigarrow \beta_1(a_1).$$

Thus, we have to find terms

$$B : \prod(n : \mathbb{N})(x : A_n),\ \beta_n(x) \rightsquigarrow \beta_1(a_1)$$
$$\Gamma : \prod(n : \mathbb{N})(x : A_n),\ \gamma_n(x) \cdot B_{n+1}(a_{n+1}) \rightsquigarrow B_n(x)$$

We will first find B using induction on the natural numbers. Suppose that $x : A_0$. A path from $\beta_0(x)$ to $\beta_1(a_1)$ is given by $\gamma_0(x)^{-1}$. Suppose that there is a term $B(n)$ of type $\prod(x : A_n),\ \beta_n(x) \rightsquigarrow \beta_1(a_1)$ and that $x : A_{n+1}$. A path from $\beta_{n+1}(x)$ to $\beta_1(a_1)$ is given by

$$B_n(a_n) \bullet \gamma_n(a_n) \bullet \gamma_{n+1}(a_{n+1}) \bullet \gamma_{n+1}(x)^{-1}$$

This gives us B. The term Γ is immediately obtained upon observing that there is a path $\gamma_n(x) \cdot B_{n+1}(a_{n+1}) \rightsquigarrow B_{n+1}(a_{n+1}) \bullet \gamma_n(x)^{-1}$ for each $x : A_n$. ∎

Theorem 3.6.4 (Invariance under homotopy). *Suppose that A is a dependent type over $\mathbb{N}$ and that $\alpha, \alpha' : \prod(n:\mathbb{N}), A_n \to A_{n+1}$ are homotopic functions. Then $[A,\alpha]_\omega \simeq [A,\alpha']_\omega$.*

PROOF. Write β and γ for the basic constructors of $[A,\alpha]_\omega$ and write β' and γ' for the basic constructors of $[A,\alpha']_\omega$. Also, let $H : \prod(n:\mathbb{N})(x:A_n),\ \alpha_n(x) \rightsquigarrow \alpha'_n(x)$ be a homotopy from α to α'.

A function $\varphi : [A,\alpha]_\omega \to [A,\alpha']_\omega$ is obtained by the induction principle for $[A,\alpha]_\omega$. For each $n:\mathbb{N}$, we take the function from $B_n :\equiv \beta'_n : A_n \to [A,\alpha']$ and for each $x : A_n$ take the path $\Gamma_n(x) :\equiv \gamma'_n(x) \bullet \beta'_{n+1}(H(x))$. This gives us the function φ for which there are paths $\mu_n(x) : \varphi(\beta_n(x)) \rightsquigarrow \beta'_n(x))$ and paths witnessing the commutativity of the diagram

$$\begin{array}{ccc}
\varphi(\beta_n(x)) & \overset{\mu_n(x)}{\rightsquigarrow} & \beta'_n(x) \\
{\scriptstyle \varphi(\gamma_n(x))}\uparrow & & \uparrow{\scriptstyle \gamma'_n(x)\bullet\beta'_{n+1}(H_n(x))} \\
\varphi(\beta_{n+1}(\alpha_n(x))) & \underset{\mu_{n+1}(\alpha_n(x))^{-1}}{\leftsquigarrow} & \beta'_{n+1}(\alpha_n(x))
\end{array}$$

for each $n:\mathbb{N}$ and each $x : A_n$. A function ψ in the other direction with paths $\mu'_n(x) : \psi(\beta'_n(x)) \rightsquigarrow \beta_n(x)$ and paths witnessing the commutativity of the diagram

$$\begin{array}{ccc}
\psi(\beta'_n(x)) & \overset{\mu'_n(x)}{\rightsquigarrow} & \beta_n(x) \\
{\scriptstyle \psi(\gamma'_n(x))}\uparrow & & \uparrow{\scriptstyle \gamma_n(x)\bullet\beta_{n+1}(H_n(x)^{-1})} \\
\psi(\beta'_{n+1}(\alpha'_n(x))) & \underset{\mu'_{n+1}(\alpha'_n(x))^{-1}}{\leftsquigarrow} & \beta_{n+1}(\alpha'_n(x))
\end{array}$$

for each $n:\mathbb{N}$ and each $x : A_n$, is obtained in a similar way. A homotopy $\psi \circ \varphi \sim \mathrm{idmap}$ can be found using the induction principle of $[A,\alpha]_\omega$. ∎

We finish this thesis with the construction of the ω-sphere and the proof that it is a contractible space. The ω-sphere is the colimit of a sequence of finite dimensional spheres, using a definition from Peter LeFanu Lumsdaine. A formal proof that the ω-sphere is contractible was first given by Guillaume Brunerie. Our approach here is a little different: where Brunerie has shown that the ω-sphere is equivalent to an ω-ball, which is a directed colimit of contractible spaces and hence contractible, we show that the inclusions of each sphere into the next are homotopic to a constant map.

Definition 3.6.5. Define the 0-sphere $\mathsf{Sphere}(0)$ as the inductive type with basic constructors

$$\mathsf{north}(0), \mathsf{south}(0) : \mathsf{Sphere}(0).$$

Thus, the 0-sphere is equivalent to the the type bool of booleans. If the n-sphere $\mathsf{Sphere}(n)$ has been defined, define the $\mathsf{n+1-Sphere}$ as the inductive type with ba-

sic constructors

$$\begin{aligned}
&\mathsf{north}(n+1) : \mathsf{Sphere}(n+1)\\
&\mathsf{south}(n+1) : \mathsf{Sphere}(n+1)\\
&\mathsf{longitude}(n) : \textstyle\prod(x : \mathsf{Sphere}(n)),\ \mathsf{north}(n+1) \rightsquigarrow \mathsf{south}(n+1).
\end{aligned}$$

The induction principle for $\mathsf{Sphere}(n+1)$ is that for every dependent type Λ over $\mathsf{Sphere}(n+1)$ for which there are

$$\begin{aligned}
&N : \Lambda(\mathsf{north}(n+1))\\
&S : \Lambda(\mathsf{south}(n+1))\\
&L : \textstyle\prod(x : \mathsf{Sphere}(n)),\ \mathsf{longitude}(n,x) \cdot N(n+1) \rightsquigarrow S(n+1)
\end{aligned}$$

there is a section $f : \prod(x : \mathsf{Sphere}(n+1)),\ \Lambda(x)$ of Λ with the property that there are paths

$$\alpha : f(\mathsf{north}(n+1)) \rightsquigarrow N \qquad \text{and} \qquad \beta : f(\mathsf{south}(n+1)) \rightsquigarrow S$$

and paths witnessing the commutativity of the square

$$\begin{array}{ccc}
f(\mathsf{south}(n+1)) & \overset{\beta}{\rightsquigarrow} & S \\
{\scriptstyle f(\mathsf{longitude}(n,x))}\uparrow & & \uparrow{\scriptstyle L(x)} \\
\mathsf{longitude}(n,x) \cdot f(\mathsf{north}(n+1)) & \underset{(\mathsf{longitude}(n,x)\cdot\alpha)^{-1}}{\leftsquigarrow} & \mathsf{longitude}(n,x) \cdot N
\end{array}$$

for each $x : \mathsf{Sphere}(n+1)$. Note that we may consider Sphere as a dependent type over $\mathbb{N}$ with sections $\mathsf{north}, \mathsf{south} : \prod(n : \mathbb{N}),\ \mathsf{Sphere}(n)$ and a function

$$\mathsf{longitude} : \textstyle\prod(n : \mathbb{N})(x : \mathsf{Sphere}(n)),\ \mathsf{north}(n+1) \rightsquigarrow \mathsf{south}(n+1). \qquad \blacklozenge$$

Before we can define S^ω we have to find maps $K_n : \mathsf{Sphere}(n) \to \mathsf{Sphere}(n+1)$ for each $n : \mathbb{N}$.

Lemma 3.6.6. *There is a function K of type*

$$\textstyle\prod(n : \mathbb{N}),\ \mathsf{Sphere}(n) \to \mathsf{Sphere}(n+1)$$

for which there are paths

$$\xi_n : K(n, \mathsf{north}(n)) \rightsquigarrow \mathsf{north}(n+1) \qquad \textit{and} \qquad \zeta_n : K(n, \mathsf{south}(n)) \rightsquigarrow \mathsf{south}(n+1)$$

for each $n : \mathbb{N}$ and a path witnessing the commutativity of the square

$$\begin{array}{ccc} K(n+1,\mathsf{south}(n+1)) & \overset{\zeta_{n+1}}{\rightsquigarrow} & \mathsf{south}(n+2) \\ \Big\uparrow{\scriptstyle K(n+1,\mathsf{longitude}(n,x))} & & \Big\uparrow{\scriptstyle \mathsf{longitude}(n+1,K(n,x))} \\ K(n+1,\mathsf{north}(n+1)) & \underset{\xi_{n+1}^{-1}}{\leftsquigarrow} & \mathsf{north}(n+2) \end{array}$$

commutes for each $n : \mathbb{N}$ and $x : \mathsf{Sphere}(n)$.

PROOF. To find $K(n)$ for each $n : \mathbb{N}$, we will first use induction on n. The function $K(0) : \mathsf{Sphere}(0) \to \mathsf{Sphere}(1)$ is given by

$$K(0,\mathsf{north}(0)) := \mathsf{north}(1) \qquad \text{and} \qquad K(0,\mathsf{south}(0)) := \mathsf{south}(1).$$

If the function $K(n)$ is given, we may construct $K(n+1)$ using the induction principle of $\mathsf{Sphere}(n+1)$. Thus, we take

$$\begin{aligned} N &:= \lambda n.\lambda x.\mathsf{north}(n+1) && : \textstyle\prod(n:\mathbb{N}),\ \mathsf{Sphere}(n) \to \mathsf{Sphere}(n+1) \\ S &:= \lambda n.\lambda x.\mathsf{south}(n+1) && : \textstyle\prod(n:\mathbb{N}),\ \mathsf{Sphere}(n) \to \mathsf{Sphere}(n+1). \end{aligned}$$

It is left to find a function

$$L : \textstyle\prod(n:\mathbb{N})(x:\mathsf{Sphere}(n)),\ \mathsf{longitude}(n,x) \cdot N(n+1) \rightsquigarrow S(n+1).$$

Note that there is a canonical path $\mathsf{longitude}(n,x) \cdot N(n+1) \rightsquigarrow N(n+1)$ for each $n : \mathbb{N}$ and each $x : \mathsf{Sphere}(n)$. By function extensionality, it therefore suffices to find a path $\mathsf{north}(n+2) \rightsquigarrow \mathsf{south}(n+2)$ for each $n : \mathbb{N}$ and each $x : \mathsf{Sphere}(n)$. We get such paths with the function

$$\lambda x.\mathsf{longitude}(n+1,K(n,x)).$$ ∎

Definition 3.6.7. The ω-sphere S^ω is defined as the directed colimit of the n-spheres using the functions $K(n)$ of the previous lemma. ♦

Lemma 3.6.8. *For any $n : \mathbb{N}$, the function $K(n)$ is homotopic to the constant function*

$$\lambda x.\mathsf{south}(n+1).$$

PROOF. The function $K(0)$ is homotopic to $\lambda x.\mathsf{south}(1)$ because there are the paths

$$\begin{aligned} \zeta_0 &: K(0,\mathsf{south}(0)) \rightsquigarrow \mathsf{south}(1) \\ \mathsf{longitude}(0,\mathsf{south}(0)) \bullet \xi_0 &: K(0,\mathsf{north}(0)) \rightsquigarrow \mathsf{south}(1). \end{aligned}$$

Suppose that $K(n) \sim \lambda x.\mathsf{south}(n+1)$. We use the induction principle of $\mathsf{Sphere}(n+1)$ to show that there is a homotopy from $K(n+1)$ to $\lambda x.\mathsf{south}(n+2)$. For the north and south poles take the paths

$$\begin{aligned} S &:\equiv \zeta_{n+1} && : K(n+1,\mathsf{south}(n+1)) \rightsquigarrow \mathsf{south}(n+2) \\ N &:\equiv \mathsf{longitude}(n+1,\mathsf{south}(n+1)) \bullet \xi_{n+1} && : K(n+1,\mathsf{north}(n+1)) \rightsquigarrow \mathsf{south}(n+2). \end{aligned}$$

It is left to find paths $L(x) : \mathsf{longitude}(n,x) \cdot N \rightsquigarrow S$ for each $x : \mathsf{Sphere}(n)$. Note that there is a canonical path

$$\mathsf{longitude}(n,x) \cdot N \rightsquigarrow N \bullet K(n+1,\mathsf{longitude}(n,x))^{-1}.$$

Now the proof is finished by noting that the square on the left in the diagram

$$\begin{array}{ccc} K(n+1,\mathsf{north}(n+1)) & \overset{\xi_{n+1}}{\rightsquigarrow} & \mathsf{north}(n+2) \\ {\scriptstyle K(n+1,\mathsf{longitude}(n,x))^{-1}}\uparrow & & \downarrow{\scriptstyle \mathsf{longitude}(n+1,K(n,x))} \quad \downarrow{\scriptstyle \mathsf{longitude}(n+1,\mathsf{south}(n+1))} \\ K(n+1,\mathsf{south}(n+1)) & \underset{\zeta_{n+1}}{\rightsquigarrow} & \mathsf{south}(n+2) \end{array}$$

commute for each $x : \mathsf{Sphere}(n)$ and that the parallel paths on the right are identical by the induction hypothesis. ∎

Theorem 3.6.9 (Brunerie). *The ω-sphere is contractible.*

PROOF. This is an immediate consequence of lemmas 3.6.3 and 3.6.8. ∎

A. THE DEPENDENT PRODUCT AS AN INDUCTIVE SPACE

Or goal in this appendix is to motivate the η-rule for dependent products from the point of view of a product space as an inductive space. Our definition of the dependent product, as we have given it in the short guide to type theory, section 1, was not as an inductive space. However, it was not too far from it either: instead of giving a universal property (the induction principle) for dependen products, we have stated the η-rule. In the following, we will observe that if we have gone the inductive way, a weaker version of the η-rule would be automatically implied. The word 'weaker' here, means that the equality of the η-rule is replaced by a path.

A note on the notation: since we cannot make use of product types while introducing them, it will be useful to use the notation of inferences.

Recall that dependent products are introduced by the rule

$$\frac{\begin{array}{rcl} & \vdash & A : \mathsf{Type} \\ x : A & \vdash & P(x) : \mathsf{Type} \end{array}}{\vdash \quad \prod(x:A),\ P(x) : \mathsf{Type}}$$

and the product $\prod(x:A),\ P(x)$ has the following canonical elements:

$$\frac{\begin{array}{rcl} & \vdash & A : \mathsf{Type} \\ x : A & \vdash & P(x) : \mathsf{Type} \\ x : A & \vdash & f(x) : P(x) \end{array}}{\vdash \quad \lambda x.f(x) : \prod(x:A),\ P(x)}$$

Instead of the usual elimination rule, which introduces a term $\mathsf{evaluate}(f,a) : P(a)$ for every $f : \prod(x:A),\ P(x)$ and $a : A$ and then asserts that there are definitional equalities

$$\mathsf{evaluate}(\lambda x.f(x), a) = f(a) \qquad (\beta)$$

$$\lambda x.\mathsf{evaluate}(f,x) = f, \qquad (\eta)$$

let's approach dependent products as inductive spaces. That is, we require that the following elimination rule holds for $\prod(x:A),\ P(x)$:

$$\frac{\begin{array}{rcl} f : \prod(x:A),\ P(x) & \vdash & D(f) : \mathsf{Type} \\ x : A,\ u(x) : P(x) & \vdash & d(x.u(x)) : D(\lambda x.u(x)) \\ & \vdash & f : \prod(x:A),\ P(x) \end{array}}{\vdash \quad R(D,d,f) : D(f)}$$

with the conversion rule

$$\frac{\begin{array}{rcl} f : \prod(x:A),\ P(x) & \vdash & D(f) : \mathsf{Type} \\ x:A,\ u(x):P(x) & \vdash & d(x.u(x)) : D(\lambda x.u(x)) \\ x:A & \vdash & u(x):P(x) \end{array}}{\vdash \quad R(D,d,\lambda x.u(x)) = d(x.u(x)) : D(\lambda x.u(x))}$$

We take this approach, because evaluate looks like an eliminator, but it lacks the strength of the typical eliminators that we see for inductive spaces. Typically, eliminators are formulated with respect to arbitrary dependent types, which is clearly not the case with evaluate. So the first thing we should do is finding out whether this inductive definition for dependent products give rise to a notion of evaluation.

To see this, suppose that $a : A$ and let $D(f)$ be the type $P(a)$ for any $f : \prod(x : A),\ P(x)$. For any inference $x : A \vdash u(x) : P(x)$ we have the term $d(x.u(x)) :\equiv u(a)$ of type $P(a)$, so the induction principle for dependent products gives a term $R(D,d,f)$ of $P(a)$ for any $f : \prod(x:A),\ P(x)$, which is precisely the term we were after. Moreover, the conversion rule states that $R(D,d,\lambda x.u(x)) = u(a)$, so we get the β-rule from the conversion rule. We will denote this $R(D,d,f)$ by $\mathsf{evaluate}(f,a)$.

If we also have identity types, we can furthermore derive the η-rule. Indeed, if $f : \prod(x:A),\ P(x)$ is a section of P then we can form the section

$$\tilde{f} :\equiv \lambda x.\mathsf{evaluate}(f,x).$$

Consider the dependent type D over $\prod(x:A),\ P(x)$ given by $D(f) :\equiv \tilde{f} \rightsquigarrow f$. Then $D(\lambda x.u(x))$ is inhabited by the identity path on $\lambda x.u(x)$ since $\mathsf{evaluate}(\lambda x.u(x),a) = u(a)$. Therefore, the induction principle for dependent products gives a path from $\tilde{f}$ to f for each $f : \prod(x:A),\ P(x)$. This means that instead of the definitional η-rule, we have the weak-η-rule

$$\lambda x.\mathsf{evaluate}(f,x) \rightsquigarrow f. \qquad \text{(weak-}\eta\text{)}$$

B. THE CIRCLE IN THE CATEGORY OF GROUPOIDS

Recall that a groupoid is a category in which every morphism is an isomorphism. The category of groupoids has small groupoids as objects and the functors between them as morphisms. We let **W**, **F** and **C** be the classes of equivalences, isofibrations and functors that are injective on objects between groupoids; it is well known that this gives the structure of a model category on the category of groupoids. Recall that a functor $\mathcal{F} : A \to B$ is an isofibration if for each $a \in A$ and each isomorphism $g : f(a) \to b'$ in B there exists an isomorphism $f : a \to a'$ such that $\mathcal{F}(f) = g$. Our aim in this section is to show that the groupoid $\mathbb{Z}$ satisfies the inductive properties of the circle. Our intuition behind this is that $\mathbb{Z}$ consists of one object $*$, which we interpret to be the base point of the circle, and the morphisms are the integers $k : * \to *$. So we may interpret loop by the integer 1.

Remark B.1. For any groupoid G, the groupoid $G^{\to}$ is the path groupoid of G. The functor i defined by $x \mapsto \mathsf{id}_x$ on objects and $g \mapsto (g, g)$ on morphisms is indeed injective on objects. It is also an equivalence, for $i \circ \mathrm{dom} \simeq \mathsf{id}_{G^{\to}}$ via the natural transformation $\tau_g = (\mathsf{id}_{\mathrm{dom}\, g}, g)$ and $\mathrm{dom} \circ i = \mathsf{id}_G$.

The functor $(\mathrm{dom}, \mathrm{cod}) : G^{\to} \to G \times G$ is an isofibration. Suppose that $g : x \to y$ is an object in $G^{\to}$ and let $(h, k) : (x, y) \to (x', y')$, then (h, k) is a morphism in $G^{\to}$ from g to $k \circ g \circ h^{-1}$ and we have clearly that $(\mathrm{dom}, \mathrm{cod})(h, k) = (h, k)$. This shows that p is indeed an isofibration. Note that $p \circ i = \Delta$, so that we may rightfully conclude that $G^{\to}$ is a path groupoid for G. ⋆

Lemma B.2. *If $f : E \to B$ is an isofibration, then every choice of an isomorphism $\Gamma(h, u) : u \to u'$ with $f(\Gamma(h, u)) = h$, for $u \in E$ and $h : f(u) \to b'$, extends to a functor*

$$\Gamma : B^{\to} \times_{\mathrm{dom}, f} E \to E^{\to}$$

with the property that $f^{\to} \circ \Gamma = \mathrm{proj}_1$. Thus, the transport functor can be defined as $\mathrm{cod} \circ \Gamma$. *We denote the transportation of u along k by $k \cdot u$.*

PROOF. Suppose that $\Gamma(h, u)$ is an isomorphism of E with domain u and with the property that $f(\Gamma(h, u)) = h$. We can extend the map $(h, u) \mapsto \Gamma(h, u)$ to a functor by defining

$$\Gamma(k, s) = (s, \Gamma(h', u') \circ s \circ \Gamma(h, u)^{-1})$$

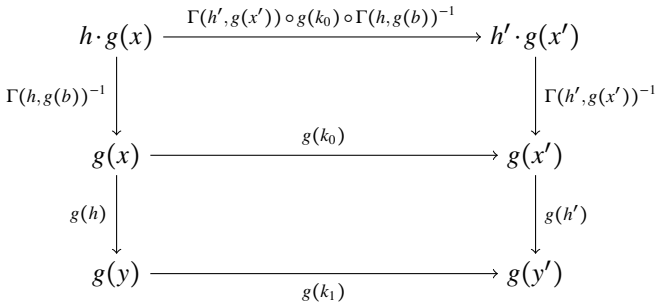
for a morphism $k : h \Rightarrow h'$ in $G^{\to}$ and a morphism $s : u \to u'$ in E. ∎

Corollary B.3. *If $g : B \to E$ is a section of f there exists a functor* $\mathrm{Map}(g) : B^{\to} \to E^{\to}$ *such that* $\mathrm{Map}(g)(h) : h \cdot g(x) \to g(y)$ *is above y whenever $h : x \to y$ is a morphism of B.*

PROOF. Define $\mathrm{Map}(g)$ on objects by $\mathrm{Map}(g)(h) = g(h) \circ \Gamma(h, g(b))^{-1}$ and on objects by

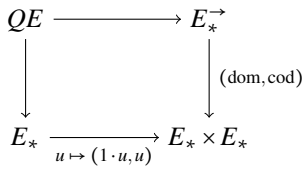
$$\mathrm{Map}(g)(k) = (\Gamma(h', g(x')) \circ g(k_0) \circ \Gamma(h, g(b))^{-1}, \mathrm{cod}(g^{\rightarrow}(k))).$$

This is illustrated by the diagram



Note that $f(\mathrm{Map}(g)(h)) = f(g(h) \circ \Gamma(h, g(b))^{-1}) = f(g(h)) \circ f(\Gamma(h, g(b)))^{-1} = h \circ h^{-1} = \mathrm{id}_y$, so that $\mathrm{Map}(g)(h)$ is indeed above y. ∎

One more ingredient is needed to show that the groupoid $\mathbb{Z}$ has the inductive properties of the circle: the groupoid of paths $p : 1 \cdot u \to u$ above 0. The fiber E_* of $*$ consists of all the objects of E and of those morphisms s of E with $f(s) = 0$. Let QE be the subgroupoid of $E_*^{\rightarrow}$ consisting of all the isomorphisms above $*$ with domain $1 \cdot u$ and codomain u, for some $u \in E_*$. So QE is defined by the pullback diagram



Note that QE consists of pairs (u, p) with $p : 1 \cdot u \to u$ in E_* as objects. The morphisms $s : (u, p) \to (v, q)$ of QE are morphisms $s : u \to v$ of E_* such that $q \circ 1 \cdot s = s \circ p$.

Lemma B.4. *There exists a functor $\mathcal{R} : QE \times \mathbb{Z} \to E$ with the property that $f \circ \mathcal{R} = \mathrm{proj}_2$, and such that the identities $\mathcal{R}(u, p, *) = u$ and $\mathrm{Map}(\mathcal{R}(u, p))(1) = p$ hold for all $(u, p) \in QE$.*

PROOF. Define R on objects by $(u, p, *) \mapsto u$ and on morphisms by $(s, k) \mapsto s \circ (p \circ \Gamma(1, u))^k$. The identity $q \circ 1 \cdot s = s \circ p$ is used to verify that $\mathcal{R}$ is indeed a functor. Then $f \circ \mathcal{R}(s, k) = f(s) + k f((p) \circ f(\Gamma(1, u))) = 0 + k(0 + 1) = k$, and hence that $f \circ \mathcal{R} = \mathrm{proj}_2$.

For the last assertion, note that

$$\begin{aligned}\mathrm{Map}(\mathcal{R}(u,p))(1) &= \mathcal{R}(u,p,1)\circ\Gamma(1,u)^{-1}\\ &= \mathcal{R}(\mathsf{id}_u,1)\circ\Gamma(1,u)^{-1}\\ &= p\circ\Gamma(1,u)\circ\Gamma(1,u)^{-1}\\ &= p.\end{aligned}$$

■

The integers have the following completely obvious non-dependent version of $\mathcal{R}$: If G is a groupoid and $p : x \to x$ is a morphism of G, then there is a functor $\mathcal{R}' : \mathbb{Z} \to G$ with $\mathcal{R}'(1) = p$.

REFERENCES

[1] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Math. Proc. Cambridge Philos. Soc.*, 146(1):45–55, 2009.

[2] Andrej Bauer and Peter LeFanu Lumsdaine. Mini-Workshop: The Homotopy Interpretation of Constructive Type Theory. *Oberwolfach Rep.*, 8(1):609–638, 2011. Abstracts from the mini-workshop held February 27–March 5, 2011, Organized by Steve Awodey, Richard Garner, Per Martin-Löf and Vladimir Voevodsky, Oberwolfach Reports. Vol. 8, no. 1.

[3] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proc. Lond. Math. Soc. (3)*, 102(2):370–394, 2011.

[4] Benno van den Berg and Richard Garner. Topological and simplicial models of identity types. *ACM Transactions on Computational Logic (TOCL)*, 13(1), 2012.

[5] Benjamin C. Pierce et al. Software foundations. `http://www.cis.upenn.edu/~bcpierce/sf/`. Course notes.

[6] Martin Hofmann. On the interpretation of type theory in locally Cartesian closed categories. In *Computer science logic (Kazimierz, 1994)*, volume 933 of *Lecture Notes in Comput. Sci.*, pages 427–441. Springer, Berlin, 1995.

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

[8] Per Martin-Löf. 100 years of Zermelo's axiom of choice: what was the problem with it? In *Logicism, intuitionism, and formalism*, volume 341 of *Synth. Libr.*, pages 209–219. Springer, Dordrecht, 2009.

[9] Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's type theory*, volume 7 of *International Series of Monographs on Computer Science*. The Clarendon Press Oxford University Press, New York, 1990. An introduction.

[10] Michael Shulman et al. `https://github.com/HoTT/HoTT`. The HoTT Coq repositories.

[11] Michael A. Warren. *Homotopy Theoretic Aspects of Constructive Type Theory*. PhD thesis, Carnegie Mellon University, 2008.