Introduction to homotopy type theory

Egbert Rijke

2019

Total number of exercises: 243

The author gratefully acknowledges the support of the Air Force Office of Scientific Research through MURI grant FA9550-15-1-0053.

Contents

Chapter I

Martin-Löf's dependent type theory

In this first chapter we explain what dependent type theory is. We begin with the structural rules of dependent type theory. The important concepts here are contexts, types in context, and terms in context, and the concept of judgmental equality. The structural rules contain rules for substitution and weakening, but they do not yet contain rules for forming new types. The informed reader may recognize that the rules we present are those of Voevodsky's *B-systems*, which are equivalent to Cartmell's *contextual categories*.

1 Dependent type theory

Dependent type theory is a system of inference rules that can be combined to make *derivations*. In these derivations, the goal is often to construct a term of a certain type. Such a term can be a function if the type of the constructed term is a function type; a proof of a property if the type of the constructed term is a proposition; an identification if the type of the constructed term is an identity type, and so on. In some respect, a type is just a collection of mathematical objects and constructing terms of a type is the everyday mathematical task or challenge. The system of inference rules that we call type theory offers a principled way of engaging in mathematical activity.

1.1 Judgments and contexts in type theory

A mathematical argument or construction consists of a sequence of deductive steps, each one using finitely many hypotheses in order to get to the next stage in the proof or construction. Such steps can be represented by **inference rules**, which are written in the form

$$\frac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \ldots \quad \mathcal{H}_n}{\mathcal{C}}$$

Inference rules containin above the horizontal line a finite list $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$ of *judgments* for the hypotheses, and below the horizontal line a single judgment $\mathcal{C}$ for the conclusion. The system of dependent type theory is described by a set of such inference rules.

A straightforward example of an inference rule that we will encounter in §2 when we introduce function types, is the inference rule

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash f : A \to B}{\Gamma \vdash f(a) : B}$$

This rule asserts that in any context Γ we may use a term $a : A$ and a function $f : A \to B$ to obtain a term $f(a) : B$. Each of the expressions

$$\Gamma \vdash a : A$$
$$\Gamma \vdash f : A \to B$$
$$\Gamma \vdash f(a) : B$$

are examples of judgments. There are four kinds of judgments in type theory:

(i) *A is a (well-formed)* ***type*** *in context Γ.* The symbolic expression for this judgment is

$$\Gamma \vdash A \text{ type}$$

(ii) *A and B are* ***judgmentally equal types*** *in context Γ.* The symbolic expression for this judgment is

$$\Gamma \vdash A \equiv B \text{ type}$$

(iii) *a is a (well-formed)* ***term*** *of type A in context Γ.* The symbolic expression for this judgment is

$$\Gamma \vdash a : A$$

(iv) *a and b are* ***judgmentally equal terms*** *of type A in context Γ.* The symbolic expression for this judgment is

$$\Gamma \vdash a \equiv b : A$$

Thus we see that any judgment is of the form $\Gamma \vdash \mathcal{J}$, consisting of a context Γ and an expression $\mathcal{J}$ asserting that A is a type, that A and B are equal types, that a is a term of type A, or that a and b are equal terms of type A. The role of a context is to declare what hypothetical terms are assumed, along with their types. More formally, a **context** is an expression of the form

$$x_1 : A_1,\ x_2 : A_2(x_1),\ \ldots,\ x_n : A_n(x_1, \ldots, x_{n-1}) \tag{1.1}$$

satisfying the condition that for each $1 \leq k \leq n$ we can derive, using the inference rules of type theory, that

$$x_1 : A_1,\ x_2 : A_2(x_1),\ \ldots,\ x_{k-1} : A_{k-1}(x_1, \ldots, x_{k-2}) \vdash A_k(x_1, \ldots, x_{k-1}) \text{ type}. \tag{1.2}$$

In other words, to check that an expression of the form Eq. (1.1) is a context, one starts on the left and works their way to the right verifying that each hypothetical term x_k is assigned a well-formed type. Hypothetical terms are commonly called **variables**, and we say that a context as in Eq. (1.1) **declares the variables** $x_1, \ldots, x_n$. We may use variable names other than $x_1, \ldots, x_n$, as long as no variable is declared more than once.

The condition in Eq. (1.2) that each of the hypothetical terms is assigned a well-formed type, is checked recursively. Note that the context of length 0 satisfies the requirement in Eq. (1.2) vacuously. This context is called the **empty context**. An expression of the form $x_1 : A_1$ is a context if and only if A_1 is a well-formed type in the empty context. Such types are called **closed types**. We will soon encounter the type $\mathbb{N}$ of natural numbers, which is an example of a closed type. There is also the notion of **closed term**, which is simply a term in the empty context. The next case is that an expression of the form $x_1 : A_1,\ x_2 : A_2(x_1)$ is a context if and only if A_1 is a well-formed type in the empty context, and $A_2(x_1)$ is a well-formed type, given a hypothetical term $x_1 : A_1$. This process repeats itself for longer contexts.

It is a feature of *dependent* type theory that all judgments are context-dependent, and indeed that even the types of the variables may depend on any previously declared variables. For example, when we introduce the *identity type* in §5, we make full use of the machinery of type dependency, as is clear from how they are introduced:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A, y : A \vdash x = y \text{ type}}$$

This rule asserts that given a type A in context Γ, we may form a type $x = y$ in context $\Gamma, x : A, y : A$. Note that in order to know that the expression $\Gamma, x : A, y : A$ is indeed a well-formed context, we need to know that A is a well-formed type in context $\Gamma, x : A$. This is an instance of *weakening*, which we will describe shortly.

In the situation where we have

$$\Gamma, x : A \vdash B(x) \text{ type},$$

we say that B is a **family** of types over A in context Γ. Alternatively, we say that $B(x)$ is a type **indexed** by $x : A$, in context Γ. Similarly, in the situation where we have

$$\Gamma, x : A \vdash b(x) : B(x),$$

we say that b is a **section** of the family B over A in context Γ. Alternatively, we say that $b(x)$ is a term of type $B(x)$, **indexed** by $x : A$ in context Γ. Note that in the above situations A, B, and b also depend on the variables declared in the context Γ, even though we have not explicitly mentioned them. It is common practice to not mention every variable in the context Γ in such situations.

1.2 Inference rules

In this section we present the basic inference rules of dependent type theory. Those rules are valid to be used in any type theoretic derivation. There are only four sets of inference rules:

(i) Rules for judgmental equality

(ii) Rules for substitution

(iii) Rules for weakening

(iv) The "variable rule"

Judgmental equality

In this set of inference rules we ensure that judgmental equality (both on types and on terms) are equivalence relations, and we make sure that in any context Γ, we can change the type of any variable to a judgmentally equal type.

The rules postulating that judgmental equality on types and on terms is an equivalence relation are as follows:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \equiv A \text{ type}} \qquad \frac{\Gamma \vdash A \equiv A' \text{ type}}{\Gamma \vdash A' \equiv A \text{ type}} \qquad \frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma \vdash A' \equiv A'' \text{ type}}{\Gamma \vdash A \equiv A'' \text{ type}}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv a' : A}{\Gamma \vdash a' \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv a' : A \qquad \Gamma \vdash a' \equiv a'' : A}{\Gamma \vdash a \equiv a'' : A}$$

Apart from the rules postulating that judgmental equality is an equivalence relation, there are also **variable conversion rules**. Informally, these are rules stating that if A and A' are judgmentally equal types in context Γ, then any valid judgment in context $\Gamma, x : A$ is also a valid judgment in context $\Gamma, x : A'$. In other words: we can convert the type of a variable to a judgmentally equal type.

The first variable conversion rule states that

$$\frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma, x : A, \Delta \vdash B(x) \text{ type}}{\Gamma, x : A', \Delta \vdash B(x) \text{ type}}$$

In this conversion rule, the context of the form $\Gamma, x : A, \Delta$ is just any extension of the context $\Gamma, x : A$, i.e., a context of the form

$$x_1 : A_1, \ldots, x_{n-1} : A_{n-1}, x : A, x_{n+1} : A_{n+1}, \ldots, x_{n+m} : A_{n+m}.$$

Similarly, there are variable conversion rules for judgmental equality of types, for terms, and for judgmental equality of terms. To avoid having to state essentially the same rule four times, we state all four variable conversion rules at once using a *generic judgment* $\mathcal{J}$, which can be any of the four kinds of judgments.

$$\frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x : A', \Delta \vdash \mathcal{J}}$$

An analogous *term conversion rule*, stated in Exercise 1.1, converting the type of a term to a judgmentally equal type, is derivable using the rules for substitution and weakening, and the variable rule.

Substitution

If we are given a term $a : A$ in context Γ, then for any type B in context $\Gamma, x : A, \Delta$ we can simultaneously substitute a for all occurences of the variable x in Δ and in B, to obtain a type $B[a/x]$ in context $\Gamma, \Delta[a/x]$. You are already familiar with simultaneous substitution, e.g., substituting 0 for x in the polynomial

$$1 + x + x^2 + x^3$$

resuts in the number $1 + 0 + 0^2 + 0^3$, which can be computed to the value 1.

Type theoretic substitution is similar. In a bit more detail, suppose we have well-formed type

$$x_1 : A_1, \ldots, x_{n-1} : A_{n-1}, x_n : A_n, x_{n+1} : A_{n+1}, \ldots, x_{n+m} : A_{n+m} \vdash B \text{ type}$$

and a term $a : A_n$ in context $x_1 : A_1, \ldots, x_{n-1} : A_{n-1}$. Then we can form the type

$$x_1 : A_1, \ldots, x_{n-1} : A_{n-1}, x_{n+1} : A_{n+1}[a/x_n], \ldots, x_{n+m} : A_{n+m}[a/x_n] \vdash B[a/x_n] \text{ type}$$

by substituting a for all occurences of x_n. Note that the variables $x_{n+1}, \ldots, x_{n+m}$ are assigned new types after performing the substitution of a for x_n.

This operation of substituting a for x is understood to be defined recursively over the length of Δ. When B is a family of types over A and $a : A$, we also say that $B[a/x]$ is the **fiber** of B at a. We will usually write $B(a)$ for $B[a/x]$. Similarly we obtain for any term $b : B$ in context $\Gamma, x : A, \Delta$ a term $b[a/x] : B[a/x]$. The term $b[a/x]$ is called the **value** of b at a. When we substitute in a judgmental equality, either of types or terms, we simply subtitute on both sides of the equation.

We can now postulate the **substitution rule** as follows:

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, \Delta[a/x] \vdash \mathcal{J}[a/x]}\ S$$

In other words, the substitution rule asserts that substitution preserves well-formedness and judgmental equality of types and terms. Furthermore, we postulate that substitution by judgmentally equal terms results in judgmentally equal types

$$\frac{\Gamma \vdash a \equiv a' : A \qquad \Gamma, x : A, \Delta \vdash B \text{ type}}{\Gamma, \Delta[a/x] \vdash B[a/x] \equiv B[a'/x] \text{ type}}$$

and it also results in judgmentally equal terms

$$\frac{\Gamma \vdash a \equiv a' : A \qquad \Gamma, x : A, \Delta \vdash b : B}{\Gamma, \Delta[a/x] \vdash b[a/x] \equiv b[a'/x] : B[a/x]}$$

To see that these rules make sense, we observe that both $B[a/x]$ and $B[a'/x]$ are types in context $\Delta[a/x]$, provided that $a \equiv a'$. This is immediate by recursion on the length of Δ.

Weakening

If we are given a type A in context Γ, then any judgment made in a longer context Γ, Δ can also be made in the context $\Gamma, x : A, \Delta$, for a fresh variable x. The **weakening rule** asserts that weakening by a type A in context preserves well-formedness and judgmental equality of types and terms.

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma, \Delta \vdash \mathcal{J}}{\Gamma, x : A, \Delta \vdash \mathcal{J}}\ W$$

This process of expanding the context by a fresh variable of type A is called **weakening** (by A).

In the simplest situation where weakening applies, we have two types A and B in context Γ. Then we can weaken B by A as follows

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}}\ W$$

in order to form the type B in context $\Gamma, x : A$. The type B in context $\Gamma, x : A$ is called the **constant family** B, or the **trivial family** B.

The variable rule

If we are given a type A in context Γ, then we can weaken A by itself to obtain that A is a type in context $\Gamma, x : A$. The **variable rule** now asserts that any hypothetical term $x : A$ in context Γ is a well-formed term of type A in context $\Gamma, x : A$.

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}\,\delta$$

One of the reasons for including the variable rule is that it provides an *identity function* on the type A in context Γ.

1.3 Derivations

A derivation in type theory is a finite tree in which each node is a valid rule of inference. At the root of the tree we find the conclusion, and in the leaves of the tree we find the hypotheses. We give two examples of derivations: a derivation showing that any variable can be changed to a fresh one, and a derivation showing that any two variables that do not mutually depend on one another can be swapped in order.

Given a derivation with hypotheses $\mathcal{H}_1, \ldots, \mathcal{H}_n$ and conclusion $\mathcal{C}$, we can form a new inference rule

$$\frac{\mathcal{H}_1 \qquad \cdots \qquad \mathcal{H}_n}{\mathcal{C}}$$

Such a rule is called **derivable**, because we have a derivation for it. In order to keep proof trees reasonably short and manageable, we use the convention that any derived rules can be used in future derivations.

Changing variables

Variables can always be changed to fresh variables. We show that this is the case by showing that the inference rule

$$\frac{\Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x' : A, \Delta[x'/x] \vdash \mathcal{J}[x'/x]}\,x'/x$$

is derivable, where x' is a variable that does not occur in the context $\Gamma, x : A, \Delta$.

Indeed, we have the following derivation using substitution, weakening, and the variable rule:

$$\frac{\dfrac{\Gamma \vdash A \text{ type}}{\Gamma, x' : A \vdash x' : A}\,\delta \qquad \dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x' : A, x : A, \Delta \vdash \mathcal{J}}\,W}{\Gamma, x' : A, \Delta[x'/x] \vdash \mathcal{J}[x'/x]}\,S$$

In this derivation it is the application of the weakening rule where we have to check that x' does not occur in the context $\Gamma, x : A, \Delta$.

Interchanging variables

The **interchange rule** states that if we have two types A and B in context Γ, and we make a judgment in context $\Gamma, x : A, y : B, \Delta$, then we can make that same judgment in context $\Gamma, y : B, x : A, \Delta$ where the order of $x : A$ and $y : B$ is swapped. More formally, the interchange rule is the following inference rule

$$\frac{\Gamma \vdash B \text{ type} \qquad \Gamma, x : A, y : B, \Delta \vdash \mathcal{J}}{\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}}$$

Just as the rule for changing variables, we claim that the interchange rule is a derivable rule.

The idea of the derivation for the interchange rule is as follows: If we have a judgment

$$\Gamma, x : A, y : B, \Delta \vdash \mathcal{J},$$

then we can change the variable y to a fresh variable y' and weaken the judgment to obtain the judgment

$$\Gamma, y : B, x : A, y' : B, \Delta[y'/y] \vdash \mathcal{J}[y'/y].$$

Now we can substitute y for y' to obtain the desired judgment $\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}$. The formal derivation is as follows:

$$\cfrac{\cfrac{\cfrac{\Gamma \vdash B \text{ type}}{\Gamma, y : B \vdash y : B}\,\delta}{\Gamma, y : B, x : A \vdash y : B}\,W \qquad \cfrac{\cfrac{\Gamma, x : A, y : B, \Delta \vdash \mathcal{J}}{\Gamma, x : A, y' : B, \Delta[y'/y] \vdash \mathcal{J}[y'/y]}\,y'/y}{\Gamma, y : B, x : A, y' : B, \Delta[y'/y] \vdash \mathcal{J}[y'/y]}\,W}{\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}}\,S$$

Exercises

1.1 Give a derivation for the following **term conversion rule**:

$$\frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma \vdash a : A}{\Gamma \vdash a : A'}$$

2 Dependent function types

A fundamental concept in dependent type theory is that of a dependent function. A dependent function is a function of which the type of the output may depend on the input. They are a generalization of ordinary functions, because an ordinary function $f : A \to B$ is a function of which the output $f(x)$ has type B regardless of the value of x.

2.1 The rules for dependent function types

Consider a section b of a family B over A in context Γ, i.e.,

$$\Gamma, x : A \vdash b(x) : B(x).$$

From one point of view, such a section b is an operation, or a program, that takes as input $x : A$ and produces a term $b(x) : B(x)$. From a more mathematical point of view we see b as a choice of an element of each $B(x)$. In other words, we may see b as a function that takes $x : A$ to

$b(x) : B(x)$. Note that the type $B(x)$ of the output is dependent on $x : A$. In this section we postulate rules for the *type* of all such dependent functions: whenever B is a family over A in context Γ, there is a type

$$\prod_{(x:A)} B(x)$$

in context Γ, consisting of all the dependent functions of which the output at $x : A$ has type $B(x)$. There are four principal rules for Π-types:

(i) The formation rule, which tells us how we may form dependent function types.

(ii) The introduction rule, which tells us how to introduce new terms of dependent function types.

(iii) The elimination rule, which tells us how to use arbitrary terms of dependent function types.

(iv) The computation rules, which tell us how the introduction and elimination rules interact. These computation rules guarantee that every term of a dependent function type behaves as expected: as a dependent function.

In the cases of the formation rule, the introduction rule, and the elimination rule, we also need rules that assert that all the constructions respect judgmental equality. Those rules are called **congruence rules**.

The Π-formation rule

The **Π-formation rule** tells us how Π-types are constructed. The idea of Π-types is that for any type family B of types over A, there is a type of dependent functions $\prod_{(x:A)} B(x)$, so the Π-formation rule is as follows:

$$\frac{\Gamma, x : A \vdash B(x) \text{ type}}{\Gamma \vdash \prod_{(x:A)} B(x) \text{ type}} \; \Pi.$$

This rule simply states that in order for the type $\prod_{(x:A)} B(x)$ to be well-formed in context Γ, the type B must be a well-formed type in context $\Gamma, x : A$.

We also require that the the operation of forming dependent function types respects judgmental equality. This is postulated in the **congruence rule** for Π-types:

$$\frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma, x : A \vdash B(x) \equiv B'(x) \text{ type}}{\Gamma \vdash \prod_{(x:A)} B(x) \equiv \prod_{(x:A')} B'(x) \text{ type}} \; \Pi\text{-eq}.$$

There is one last rule that we need about the formation of Π-types, asserting that it does not matter what name we use for the variable x that appears in the expression $\prod_{(x:A)} B(x)$. More precisely, when x' is a variable that does not occur in the context Γ, then we postulate that

$$\frac{\Gamma, x : A \vdash B(x) \text{ type}}{\Gamma \vdash \prod_{(x:A)} B(x) \equiv \prod_{(x':A)} B(x') \text{ type}} \; \Pi\text{-}x'/x.$$

This rule is also known as **α-conversion** for Π-types.

The Π-introduction rule

The introduction rule for dependent functions tells us how we may construct dependent functions of type $\prod_{(x:A)} B(x)$. The idea is that a dependent function $f : \prod_{(x:A)} B(x)$ is an operation that takes an $x : A$ to $f(x) : B(x)$. Hence the introduction rule of dependent functions postulates that, in order to construct a dependent function one has to construct a term $b(x) : B(x)$ in context $x : A$, i.e.:

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x.\, b(x) : \prod_{(x:A)} B(x)}\ \lambda.$$

This introduction rule for dependent functions is also called the **λ-abstraction rule**, and we also say that the λ-abstraction $\lambda x.\, b(x)$ **binds** the variable x in b. Just like ordinary mathematicians, we will sometimes write $x \mapsto b(x)$ for a function $\lambda x.\, b(x)$. The map $n \mapsto n^2$ is an example.

The λ-abstraction is also required to respect judgmental equality. Therefore we postulate the **congruence rule** for λ-abstraction, which asserts that

$$\frac{\Gamma, x : A \vdash b(x) \equiv b'(x) : B(x)}{\Gamma \vdash \lambda x.\, b(x) \equiv \lambda x.\, b'(x) : \prod_{(x:A)} B(x)}\ \lambda\text{-eq}.$$

The Π-elimination rule

The elimination rule for dependent function types provides us with a way to *use* dependent functions. The way to use a dependent function is to apply it to an argument of the domain type. The Π-elimination rule is therefore also called the **evaluation rule**. It asserts that given a dependent function $f : \prod_{(x:A)} B(x)$ in context Γ we obtain a term $f(x)$ of type $B(x)$ in context $\Gamma, x : A$. More formally:

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma, x : A \vdash f(x) : B(x)}\ ev$$

Again we require that evaluation respects judgmental equality:

$$\frac{\Gamma \vdash f \equiv f' : \prod_{(x:A)} B(x)}{\Gamma, x : A \vdash f(x) \equiv f'(x) : B(x)}$$

The Π-computation rules

We now postulate rules that specify the behavior of functions. First, we have a rule that asserts that a function of the form $\lambda x.\, b(x)$ behaves as expected: when we evaluate it at $x : A$, then we obtain the value $b(x) : B(x)$. This rule is called the **β-rule**

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma, x : A \vdash (\lambda y.b(y))(x) \equiv b(x) : B(x)}\ \beta.$$

Second, we postulate a rule that asserts that all elements of a Π-type are (dependent) functions. This rule is known as the **η-rule**

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma \vdash \lambda x.\, f(x) \equiv f : \prod_{(x:A)} B(x)}\ \eta.$$

In other words, the computation rules (β and η) for dependent function types postulate that λ-abstraction rule and the evaluation rule are mutual inverses. This completes the specification of dependent function types.

2.2 Ordinary function types

Given two types A and B in context Γ, we can use the rules for Π-types to form the type $A \to B$ of *ordinary* functions from A to B. The type of ordinary functions is obtained by first weakening B by A and subsequently applying the Π-formation rule, as in the following derivation:

$$\dfrac{\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}}\, W}{\Gamma \vdash \prod_{(x:A)} B \text{ type}}\, \Pi$$

A term $f : \prod_{(x:A)} B$ is a function that takes an argument $x : A$ and returns $f(x) : B$. In other words, terms of type $\prod_{(x:A)} B$ are indeed ordinary functions from A to B. Therefore we will write $A \to B$ for the **type of functions** from A to B. Sometimes we will also write B^A for the type $A \to B$.

We give a brief summary of the rules specifying ordinary function types, omitting the congruence rules. All of these rules can be derived easily from the corresponding rules for Π-types.

$$\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \to B \text{ type}}\, \to$$

$$\dfrac{\Gamma \vdash B \text{ type} \qquad \Gamma, x : A \vdash b(x) : B}{\Gamma \vdash \lambda x.\, b(x) : A \to B}\, \lambda \qquad \dfrac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash f(x) : B}\, ev$$

$$\dfrac{\Gamma \vdash B \text{ type} \qquad \Gamma, x : A \vdash b(x) : B}{\Gamma, x : A \vdash (\lambda y.\, b(y))(x) \equiv b(x) : B}\, \beta \qquad \dfrac{\Gamma \vdash f : A \to B}{\Gamma \vdash \lambda x.\, f(x) \equiv f : A \to B}\, \eta$$

2.3 The identity function, composition, and their laws

First, we use the rules of dependent type theory to construct the identity function on an arbitrary type.

Definition 2.3.1. For any type A in context Γ, we define the **identity function** $\mathsf{id}_A : A \to A$ using the variable rule:

$$\dfrac{\dfrac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}}{\Gamma \vdash \mathsf{id}_A :\equiv \lambda x.\, x : A \to A}$$

Note that we have used the symbol $:\equiv$ in the conclusion to define the identity function. A judgment of the form $\Gamma \vdash a :\equiv b : A$ should be read as "b is a well-defined term of type A in context Γ, and we will refer to it as a".

By the above construction of the identity function we see that the identity function can be introduced with the following rule

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \mathsf{id}_A : A \to A}$$

Moreover, by the β-rule we see that the identity function satisfies the following computation rule:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{id}_A(x) \equiv x : A}$$

Next, we define the composition of functions. We will introduce the composition operation itself as a function comp that takes two arguments: the first argument is a function $g : B \to C$, and the second argument is a function $f : A \to B$. The output is a function $\mathsf{comp}(g, f) : A \to C$, for which we often write $g \circ f$.

Types of functions with multiple arguments can be formed by iterating the Π-formation rule or the $\to$-formation rule. For example, a function

$$f : A \to (B \to C)$$

takes two arguments: first it takes an argument $x : A$, and the output $f(x)$ has type $B \to C$. This is again a function type, so $f(x)$ is a function that takes an argument $y : B$, and its output $f(x)(y)$ has type C. We will usually write $f(x, y)$ for $f(x)(y)$. With this idea of iterating function types, we see that type of the composition operation comp should be

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

It is the type of functions, taking a function $g : B \to C$, to the type of functions $(A \to B) \to (A \to C)$. Thus, $\mathsf{comp}(g)$ is again a function, mapping a function $f : A \to B$ to the type of functions $B \to C$.

Definition 2.3.2. For any three types A, B, and C in context Γ, there is a **composition** operation

$$\mathsf{comp} : (B \to C) \to ((A \to B) \to (A \to C)),$$

i.e., we can derive

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash C \text{ type}}{\Gamma \vdash \mathsf{comp} : (B \to C) \to ((A \to B) \to (A \to C))}$$

We will usually write $g \circ f$ for $\mathsf{comp}(g, f)$. Moreover, the composition operation satisfies the following computation rule:

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash C \text{ type}}{\Gamma, g : B \to C, f : A \to B, x : A \vdash \mathsf{comp}(g, f, x) \equiv g(f(x)) : C}$$

Construction. The idea of the definition is to define $\mathsf{comp}(g, f)$ to be the function $\lambda x.\, g(f(x))$. The derivation we use to construct comp is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, f : B^A, x : A \vdash f(x) : B}}{\Gamma, g : C^B, f : B^A, x : A \vdash f(x) : B} \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash B \text{ type} \qquad \Gamma \vdash C \text{ type}}{\Gamma, g : C^B, y : B \vdash g(y) : C}}{\Gamma, g : C^B, f : B^A, y : B \vdash g(y) : C}}{\Gamma, g : C^B, f : B^A, x : A, y : B \vdash g(y) : C}}{\Gamma, g : C^B, f : B^A, x : A \vdash g(f(x)) : C}}{\Gamma, g : C^B, f : B^A \vdash \lambda x.\, g(f(x)) : C^A}}{\Gamma, g : B \to C \vdash \lambda f.\, \lambda x.\, g(f(x)) : B^A \to C^A}}{\Gamma \vdash \mathsf{comp} :\equiv \lambda g.\, \lambda f.\, \lambda x.\, g(f(x)) : C^B \to (B^A \to C^A)}$$

It is immediate by the β-rule that the composition operation satisfies the asserted computation rule. ☐

The rules of function types can be used to derive the laws of a category for functions, i.e., we can derive that function composition is associative and that the identity function satisfies the unit laws. In the remainder of this section we will give these derivations.

Lemma 2.3.3. *Composition of functions is associative, i.e., we can derive*

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash g : B \to C \qquad \Gamma \vdash h : C \to D}{\Gamma \vdash (h \circ g) \circ f \equiv h \circ (g \circ f) : A \to D}$$

Proof. The main idea of the proof is that both $((h \circ g) \circ f)(x)$ and $(h \circ (g \circ f))(x)$ evaluate to $h(g(f(x)))$, and therefore $(h \circ g) \circ f$ and $h \circ (g \circ f)$ must be judgmentally equal. This idea is made formal in the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash f(x) : B} \qquad \dfrac{\dfrac{\Gamma \vdash g : B \to C}{\Gamma, y : B \vdash g(y) : C}}{\Gamma, x : A, y : B \vdash g(y) : C}}{\Gamma, x : A \vdash g(f(x)) : C} \qquad \dfrac{\dfrac{\Gamma \vdash h : C \to D}{\Gamma, z : C \vdash h(z) : D}}{\Gamma, x : A, z : C \vdash h(z) : D}}{\Gamma, x : A \vdash h(g(f(x))) : D}}{\Gamma, x : A \vdash h(g(f(x))) \equiv h(g(f(x))) : D}}{\Gamma, x : A \vdash (h \circ g)(f(x)) \equiv h((g \circ f)(x)) : D}}{\Gamma, x : A \vdash ((h \circ g) \circ f)(x) \equiv (h \circ (g \circ f))(x) : D}}{\Gamma \vdash (h \circ g) \circ f \equiv h \circ (g \circ f) : A \to D.}$$

☐

Lemma 2.3.4. *Composition of functions satisfies the left and right unit laws, i.e., we can derive*

$$\frac{\Gamma \vdash f : A \to B}{\Gamma \vdash \mathsf{id}_B \circ f \equiv f : A \to B}$$

and

$$\frac{\Gamma \vdash f : A \to B}{\Gamma \vdash f \circ \mathsf{id}_A \equiv f : A \to B}$$

Proof. The derivation for the left unit law is

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash f(x) : B} \qquad \dfrac{\Gamma \vdash A\ \text{type} \qquad \dfrac{\Gamma \vdash B\ \text{type}}{\Gamma, y : B \vdash \mathsf{id}(y) \equiv y : B}}{\Gamma, x : A, y : B \vdash \mathsf{id}(y) \equiv y : B}}{\Gamma, x : A \vdash \mathsf{id}(f(x)) \equiv f(x) : B}}{\Gamma \vdash \lambda x.\, \mathsf{id}(f(x)) \equiv \lambda x.\, f(x) : A \to B} \qquad \dfrac{\Gamma \vdash f : A \to B}{\Gamma \vdash \lambda x.\, f(x) \equiv f : A \to B}}{\Gamma \vdash \mathsf{id} \circ f \equiv f : A \to B}$$

The right unit law is left as Exercise 2.2. ☐

Exercises

2.1 The η-rule is often seen as an extensionality principle. Use the η-rule to show that if f and g take equal values, then they must be equal, i.e., give a derivation for the rule

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x) \qquad \Gamma \vdash g : \prod_{(x:A)} B(x) \qquad \Gamma, x : A \vdash f(x) \equiv g(x) : B(x)}{\Gamma \vdash f \equiv g : \prod_{(x:A)} B(x)}$$

2.2 Give a derivation for the right unit law of Lemma 2.3.4.

2.3 Show that the rule

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x.\, b(x) \equiv \lambda x'.\, b(x') : \prod_{(x:A)} B(x)}\ \lambda\text{-}x'/x$$

is derivable for any variable x' that does not occur in the context $\Gamma, x : A$.

2.4 (a) Construct the **constant function**

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, y : B \vdash \mathsf{const}_y : A \to B}$$

(b) Show that

$$\frac{\Gamma \vdash f : A \to B}{\Gamma, z : C \vdash \mathsf{const}_z \circ f \equiv \mathsf{const}_z : A \to C}$$

(c) Show that

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash g : B \to C}{\Gamma, y : B \vdash g \circ \mathsf{const}_y \equiv \mathsf{const}_{g(y)} : A \to C}$$

2.5 (a) Given two types A and B in context Γ, and a type C in context $\Gamma, x : A, y : B$, define the **swap function**

$$\Gamma \vdash \sigma : \Big(\prod_{(x:A)} \prod_{(y:B)} C(x,y)\Big) \to \Big(\prod_{(y:B)} \prod_{(x:A)} C(x,y)\Big)$$

that swaps the order of the arguments.

(b) Show that

$$\Gamma \vdash \sigma \circ \sigma \equiv \mathsf{id} : \Big(\prod_{(x:A)} \prod_{(y:B)} C(x,y)\Big) \to \Big(\prod_{(x:A)} \prod_{(y:B)} C(x,y)\Big).$$

3 The natural numbers

The set of natural numbers is the most important object in mathematics. We quote Bishop, from his Constructivist Manifesto, the first chapter in Foundations of Constructive Analysis [2], where he gives a colorful illustration of its importance to mathematics.

> "The primary concern of mathematics is number, and this means the positive integers. We feel about number the way Kant felt about space. The positive integers and their arithmetic are presupposed by the very nature of our intelligence and, we are tempted to believe, by the very nature of intelligence in general. The development of the theory of the positive integers from the primitive concept of the unit, the concept of adjoining a unit, and the process of mathematical induction carries complete conviction. In the

> words of Kronecker, the positive integers were created by God. Kronecker would have expressed it even better if he had said that the positive integers were created by God for the benefit of man (and other finite beings). Mathematics belongs to man, not to God. We are not interested in properties of the positive integers that have no descriptive meaning for finite man. When a man proves a positive integer to exist, he should show how to find it. If God has mathematics of his own that needs to be done, let him do it himself."

A bit later in the same chapter, he continues:

> "Building on the positive integers, weaving a web of ever more sets and ever more functions, we get the basic structures of mathematics: the rational number system, the real number system, the euclidean spaces, the complex number system, the algebraic number fields, Hilbert space, the classical groups, and so forth. Within the framework of these structures, most mathematics is done. Everything attaches itself to number, and every mathematical statement ultimately expresses the fact that if we perform certain computations within the set of positive integers, we shall get certain results."

3.1 The formal specification of the type of natural numbers

The type $\mathbb{N}$ of **natural numbers** is the archetypal example of an inductive type. The rules we postulate for the type of natural numbers come in four sets, just as the rules for Π-types:

(i) The formation rule, which asserts that the type $\mathbb{N}$ can be formed.

(ii) The introduction rules, which provide the zero element and the successor function.

(iii) The elimination rule. This rule is the type theoretic analogue of the induction principle for $\mathbb{N}$.

(iv) The computation rules, which assert that any application of the elimination rule behaves as expected on the constructors $0_{\mathbb{N}}$ and $\mathsf{succ}_{\mathbb{N}}$ of $\mathbb{N}$.

The formation rule of $\mathbb{N}$

The type $\mathbb{N}$ is formed by the $\mathbb{N}$**-formation** rule

$$\frac{}{\vdash \mathbb{N}\ \text{type.}}\ \mathbb{N}\text{-form}$$

In other words, $\mathbb{N}$ is postulated to be a closed type.

The introduction rules of $\mathbb{N}$

Unlike the set of positive integers in Bishop's remarks, Peano's first axiom postulates that 0 is a natural number. The introduction rules for $\mathbb{N}$ equip it with the **zero term** and the **successor function**.

$$\frac{}{\vdash 0_{\mathbb{N}} : \mathbb{N}} \qquad\qquad \frac{}{\vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}$$

Remark 3.1.1. We annotate the terms $0_{\mathbb{N}}$ and $\mathsf{succ}_{\mathbb{N}}$ of type $\mathbb{N}$ with their type in the subscript, as a reminder that $0_{\mathbb{N}}$ and $\mathsf{succ}_{\mathbb{N}}$ are declared to be terms of type $\mathbb{N}$, and not of any other type. In the next chapter we will introduce the type $\mathbb{Z}$ of the integers, on which we can also define a zero term $0_{\mathbb{Z}}$, and a successor function $\mathsf{succ}_{\mathbb{Z}}$. These should be distinguished from the terms $0_{\mathbb{N}}$ and $\mathsf{succ}_{\mathbb{N}}$. In general, we will make sure that every term is given a unique name. In libraries of mathematics formalized in a computer proof assistant it is also the case that every type must be given a unique name.

The elimination rule of $\mathbb{N}$

To prove properties about the natural numbers, we postulate an *induction principle* for $\mathbb{N}$. For a typical example, it is easy to show by induction that

$$1 + \cdots + n = \frac{n(n+1)}{2}.$$

Similarly, we can define operations by recursion on the natural numbers: the Fibonacci sequence is defined by $F(0) = 0$, $F(1) = 1$, and

$$F(n+2) = F(n) + F(n+1).$$

Needless to say, we want an induction principle to hold for the natural numbers in type theory and we also want it to be possible to construct operations on the natural numbers by recursion.

In dependent type theory we may think of a type family P over $\mathbb{N}$ as a *predicate* over $\mathbb{N}$. Especially after we introduce a few more type-forming operations, such as Σ-types and identity types, it will become clear that the language of dependent type theory expressive enough to find definitions of all of the standard concepts and operations of elementary number theory in type theory. Many of those definitions, the ordering relations $\leq$ and $<$ for example, will make use of type dependency. Then, to prove that $P(n)$ 'holds' for all n we just have to construct a dependent function

$$\prod_{(n:\mathbb{N})} P(n).$$

The induction principle for the natural numbers in type theory exactly states what one has to do in order to construct such a dependent function, via the following inference rule:

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n)\ \text{type} \\ \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_S : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S) : \prod_{(n:\mathbb{N})} P(n)}\ \mathbb{N}\text{-ind}$$

Just like for the usual induction principle of the natural numbers, there are two things to be constructed given a type family P over $\mathbb{N}$: in the **base case** we need to construct a term $p_0 : P(0_{\mathbb{N}})$, and for the **inductive step** we need to construct a function of type $P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))$ for all $n : \mathbb{N}$. And this comes at one immediate advantage: induction and recursion in type theory are one and the same thing!

Remark 3.1.2. We might alternatively present the induction principle of $\mathbb{N}$ as the following inference rule

$$\frac{\Gamma, n : \mathbb{N} \vdash P(n)\ \text{type}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}} : P(0_{\mathbb{N}}) \to \Big(\Big(\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\Big) \to \prod_{(n:\mathbb{N})} P(n)\Big)}$$

In other words, for any type family P over $\mathbb{N}$ there is a *function* $\mathsf{ind}_{\mathbb{N}}$ that takes two arguments, one for the base case and one for the inductive step, and returns a section of P. Now it is justified to wonder: is this slightly different presentation of induction equivalent to the previous presentation?

To see that indeed we get such a function from the induction principle (rule $\mathbb{N}$-ind above), we note that the induction principle is stated to hold in an *arbitrary* context Γ. So let us wield the power of type dependency: by weakening and the variable rule we have the following well-formed terms:

$$\Gamma,\ p_0 : P(0_{\mathbb{N}}),\ p_S : \textstyle\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n)) \vdash p_0 : P(0_{\mathbb{N}})$$
$$\Gamma,\ p_0 : P(0_{\mathbb{N}}),\ p_S : \textstyle\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n)) \vdash p_S : \textstyle\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n)).$$

Therefore, the induction principle of $\mathbb{N}$ provides us with a term

$$\Gamma,\ p_0 : P(0_{\mathbb{N}}),\ p_S : \textstyle\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n)) \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S) : \textstyle\prod_{(n:\mathbb{N})} P(n).$$

By λ-abstraction we now obtain a function

$$\mathsf{ind}_{\mathbb{N}} : P(0_{\mathbb{N}}) \to \Big(\Big(\textstyle\prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\Big) \to \textstyle\prod_{(n:\mathbb{N})} P(n)\Big)$$

in context Γ. Therefore we see that it does not really matter whether we present the induction principle of $\mathbb{N}$ in a more verbose way as an inference rule with the base case and the inductive step as hypotheses, or as a function taking variables for the base case and the inductive step as arguments.

The computation rules of $\mathbb{N}$

The **computation rules** for $\mathbb{N}$ postulate that the dependent function $\mathsf{ind}_{\mathbb{N}}(P, p_0, p_S)$ behaves as expected when it is applied to $0_{\mathbb{N}}$ or a successor. There is one computation rule for each step in the induction principle, covering the base case and the inductive step.

The computation rule for the base case is

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n)\ \mathrm{type} \\ \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_S : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, 0_{\mathbb{N}}) \equiv p_0 : P(0_{\mathbb{N}})}$$

Similarly, with the same hypotheses as for the computation rule for the base case, the computation rule for the inductive step is

$$\frac{\dots}{\Gamma, n : \mathbb{N} \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, \mathsf{succ}_{\mathbb{N}}(n)) \equiv p_S(n, \mathsf{ind}_{\mathbb{N}}(p_0, p_S, n)) : P(\mathsf{succ}_{\mathbb{N}}(n))}$$

This completes the formal specification of $\mathbb{N}$.

3.2 Addition on the natural numbers

Using the induction principle of $\mathbb{N}$ we can perform many familiar constructions. For instance, we can define the **addition operation** by induction on $\mathbb{N}$.

Definition 3.2.1. We define a function

$$\mathsf{add}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

satisfying $\mathsf{add}_{\mathbb{N}}(0_{\mathbb{N}}, n) \equiv n$ and $\mathsf{add}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(m), n) \equiv \mathsf{succ}_{\mathbb{N}}(\mathsf{add}_{\mathbb{N}}(m, n))$. Usually we will write $n + m$ for $\mathsf{add}_{\mathbb{N}}(n, m)$.

We first give an informal construction of the addition operation, explaining the ideas behind the construction. This is important, because there are many binary operations on the natural numbers. The correctness of a formal construction of a term

$$\vdash \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

only shows us that we have correctly constructed a binary operation on the natural numbers, but this doesn't tell us that the operation we've defined is deserving of the name addition. There are indeed many binary operations on the natural numbers, such as the $\mathsf{min}_{\mathbb{N}}$, $\mathsf{max}_{\mathbb{N}}$, and multiplication operations, so we need to be careful to make sure that the binary operation we are constructing really is the addition operation.

Informal construction. Our goal is to construct a function of type

$$\vdash \mathsf{add}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}).$$

By λ-abstraction it therefore suffices to construct a term

$$m : \mathbb{N} \vdash \mathsf{add}_{\mathbb{N}}(m) : \mathbb{N} \to \mathbb{N}.$$

Such a term is constructed by induction. Since we are defining addition, we want our definition of $\mathsf{add}_{\mathbb{N}}$ to be such that

$$\begin{aligned} \mathsf{add}_{\mathbb{N}}(m, 0_{\mathbb{N}}) &\equiv n \\ \mathsf{add}_{\mathbb{N}}(m, \mathsf{succ}_{\mathbb{N}}(n)) &\equiv \mathsf{succ}_{\mathbb{N}}(\mathsf{add}_{\mathbb{N}}(m, n)). \end{aligned}$$

In other words, our definition of addition is such that $m + 0 \equiv m$ and $m + \mathsf{succ}_{\mathbb{N}}(n) \equiv \mathsf{succ}_{\mathbb{N}}(m + n)$.

The inductive proof requires us to define a term

$$n : \mathbb{N} \vdash \mathsf{add\text{-}zero}_{\mathbb{N}}(n) :\equiv n : \mathbb{N}$$

in the base case, and a term

$$n : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(n) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

in the inductive step. The result of the inductive proof will then be a function $\mathsf{add}_{\mathbb{N}}(n) : \mathbb{N} \to \mathbb{N}$ satisfying

$$\begin{aligned} n : \mathbb{N} &\vdash \mathsf{add}_{\mathbb{N}}(n, 0_{\mathbb{N}}) \equiv \mathsf{add\text{-}zero}_{\mathbb{N}}(n) : \mathbb{N} \\ n : \mathbb{N} &\vdash \mathsf{add}_{\mathbb{N}}(n, \mathsf{succ}_{\mathbb{N}}(m)) \equiv \mathsf{add\text{-}succ}_{\mathbb{N}}(n, m, \mathsf{add}_{\mathbb{N}}(n, m)). \end{aligned}$$

Anticipating these computation rules, we see that the following choices result in an addition operation with the expected behavior:

$$\begin{aligned} n : \mathbb{N} &\vdash \mathsf{add\text{-}zero}_{\mathbb{N}}(n) :\equiv n : \mathbb{N} \\ n : \mathbb{N} &\vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(n) :\equiv \mathsf{const}_{\mathsf{succ}_{\mathbb{N}}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}). \end{aligned}$$ □

Formal derivation. The derivation for the construction of $\mathsf{add\text{-}succ}_{\mathbb{N}}$ looks as follows:

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathbb{N}\ \text{type}} \qquad \dfrac{\dfrac{}{\vdash \mathbb{N}\ \text{type}} \qquad \dfrac{}{\vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{x : \mathbb{N} \vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{n : \mathbb{N}, x : \mathbb{N} \vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{n : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(n) :\equiv \lambda x.\, \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}$$

We combine this derivation with the induction principle of $\mathbb{N}$ to complete the construction of addition:

$$\dfrac{\dfrac{\vdots}{n : \mathbb{N} \vdash \mathsf{add\text{-}zero}_{\mathbb{N}}(n) :\equiv n : \mathbb{N}} \qquad \dfrac{\vdots}{n : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(n) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}}{n : \mathbb{N} \vdash \mathsf{add}_{\mathbb{N}}(n) \equiv \mathsf{ind}_{\mathbb{N}}(\mathsf{add\text{-}zero}_{\mathbb{N}}, \mathsf{add\text{-}succ}_{\mathbb{N}}) : \mathbb{N} \to \mathbb{N}}$$

The asserted judgmental equalities then hold by the computation rules for $\mathbb{N}$. □

Remark 3.2.2. When we define a function $f : \prod_{(n:\mathbb{N})} P(n)$, we will often do so just by indicating its definition on $0_{\mathbb{N}}$ and its definition on $\mathsf{succ}_{\mathbb{N}}(n)$, by writing

$$\begin{aligned} f(0_{\mathbb{N}}) &:\equiv p_0 \\ f(\mathsf{succ}_{\mathbb{N}}(n)) &:\equiv p_S(n, f(n)). \end{aligned}$$

For example, the definition of addition on the natural numbers could be given as

$$\begin{aligned} \mathsf{add}_{\mathbb{N}}(0_{\mathbb{N}}, n) &:\equiv n \\ \mathsf{add}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(m), n) &:\equiv \mathsf{succ}_{\mathbb{N}}(\mathsf{add}_{\mathbb{N}}(m, n)). \end{aligned}$$

This way of defining a function is called *pattern matching*. A more formal inductive argument can be obtained from a definition by pattern matching if it is possible to obtain from the expression $p_S(n, f(n))$ a general dependent function

$$p_S : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n)).$$

In practice this is usually the case. Computer proof assistants such as Agda have sophisticated algorithms to allow for definitions by pattern matching.

Remark 3.2.3. By the computation rules for $\mathbb{N}$ it follows that

$$m + 0_{\mathbb{N}} \equiv m, \qquad \text{and} \qquad m + \mathsf{succ}_{\mathbb{N}}(n) \equiv \mathsf{succ}_{\mathbb{N}}(m + n).$$

A simple consequence of this definition is that $\mathsf{succ}_{\mathbb{N}}(n) \equiv n + 1$, as one would expect. However, the rules that we provided so far are not sufficient to also conclude that $0_{\mathbb{N}} + n \equiv n$ and $\mathsf{succ}_{\mathbb{N}}(m) + n \equiv \mathsf{succ}_{\mathbb{N}}(m + n)$. In fact, we will not be able to prove such judgmental equalities. Nevertheless, once we have introduced the *identity type* in §5 we will be able to *identify* $0_{\mathbb{N}} + n$ with n, and $\mathsf{succ}_{\mathbb{N}}(m) + n$ with $\mathsf{succ}_{\mathbb{N}}(m + n)$. See Exercise 5.5.

Exercises

3.1 Define the binary **min** and **max** functions

$$\mathsf{min}_{\mathbb{N}}, \mathsf{max}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}).$$

3.2 Define the **multiplication** operation

$$\mathsf{mul}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}).$$

3.3 Define the **exponentiation function** $n, m \mapsto m^n$ of type $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.
3.4 Define the **factorial** operation $n \mapsto n!$.
3.5 Define the **binomial coefficient** $\binom{n}{k}$ for any $n, k : \mathbb{N}$, making sure that $\binom{n}{k} \equiv 0$ when $n < k$.

3.6 Use the induction principle of $\mathbb{N}$ to define the **Fibonacci sequence** as a function $F : \mathbb{N} \to \mathbb{N}$ that satisfies the equations

$$\begin{aligned} F(0_{\mathbb{N}}) &\equiv 0_{\mathbb{N}} \\ F(1_{\mathbb{N}}) &\equiv 1_{\mathbb{N}} \\ F(\mathsf{succ}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n))) &\equiv F(n) + F(\mathsf{succ}_{\mathbb{N}}(n)). \end{aligned}$$

4 More inductive types

Analogous to the type of natural numbers, many types can be specified as inductive types. In this section we introduce some further examples of inductive types: the unit type, the empty type, the booleans, coproducts, dependent pair types, and cartesian products. We also introduce the type of integers.

4.1 The idea of general inductive types

Just like the type of natural numbers, other inductive types are also specified by their *constructors*, an *induction principle*, and their *computation rules*:

(i) The constructors tell what structure the inductive type comes equipped with. There may any finite number of constructors, even no constructors at all, in the specification of an inductive type.

(ii) The induction principle specifies the data that should be provided in order to construct a section of an arbitrary type family over the inductive type.

(iii) The computation rules assert that the inductively defined section agrees on the constructors with the data that was used to define the section. Thus, there is a computation rule for every constructor.

The induction principle and computation rules can be generated automatically once the constructors are specified, but it goes beyond the scope of our course to describe general inductive types.

4.2 The unit type

A straightforward example of an inductive type is the *unit type*, which has just one constructor. Its induction principle is analogous to just the base case of induction on the natural numbers.

Definition 4.2.1. We define the **unit type** to be a closed type $\mathbf{1}$ equipped with a closed term

$$\star : \mathbf{1},$$

satisfying the induction principle that for any type family of types $P(x)$ indexed by $x : \mathbf{1}$, there is a term

$$\mathsf{ind}_{\mathbf{1}} : P(\star) \to \prod_{(x:\mathbf{1})} P(x)$$

for which the computation rule

$$\mathsf{ind}_{\mathbf{1}}(p, \star) \equiv p$$

holds. Sometimes we write $\lambda \star .\, p$ for $\mathsf{ind}_{\mathbf{1}}(p)$.

The induction principle can also be used to define ordinary functions out of the unit type. Indeed, given a type A we can first weaken it to obtain the constant family over $\mathbf{1}$, with value A. Then the induction principle of the unit type provides a function

$$\mathsf{ind}_{\mathbf{1}} : A \to (\mathbf{1} \to A).$$

In other words, by the induction principle for the unit type we obtain for every $x : A$ a function $\mathsf{pt}_x :\equiv \mathsf{ind}_{\mathbf{1}}(x) : \mathbf{1} \to A$.

4.3 The empty type

The empty type is a degenerate example of an inductive type. It does *not* come equipped with any constructors, and therefore there are also no computation rules. The induction principle merely asserts that any type family has a section. In other words: if we assume the empty type has a term, then we can prove anything.

Definition 4.3.1. We define the **empty type** to be a type $\emptyset$ satisfying the induction principle that for any family of types $P(x)$ indexed by $x :$, there is a term

$$\mathsf{ind}_{\emptyset} : \prod_{(x:\emptyset)} P(x).$$

The induction principle for the empty type can also be used to construct a function

$$\emptyset \to A$$

for any type A. Indeed, to obtain this function one first weakens A to obtain the constant family over $\emptyset$ with value A, and then the induction principle gives the desired function.

Thus we see that from the empty type anything follows. Therefore, we we see that anything follows from A, if we have a function from A to the empty type. This motivates the following definition.

Definition 4.3.2. For any type A we define **negation** of A by

$$\neg A :\equiv A \to \emptyset.$$

Since $\neg A$ is the type of functions from A to $\emptyset$, a proof of $\neg A$ is given by assuming that A holds, and then deriving a contradiction. This proof technique is called **proof of negation**. Proofs of negation are not to be confused with *proofs by contradiction*. In type theory there is no way of obtaining a term of type A from a term of type $(A \to \emptyset) \to \emptyset$.

4.4 The booleans

Definition 4.4.1. We define the **booleans** to be a type $\mathbf{2}$ that comes equipped with

$$0_{\mathbf{2}} : \mathbf{2}$$
$$1_{\mathbf{2}} : \mathbf{2}$$

satisfying the induction principle that for any family of types $P(x)$ indexed by $x : \mathbf{2}$, there is a term

$$\mathsf{ind}_{\mathbf{2}} : P(0_{\mathbf{2}}) \to \Big(P(1_{\mathbf{2}}) \to \prod_{(x:\mathbf{2})} P(x)\Big)$$

for which the computation rules

$$\mathsf{ind}_{\mathbf{2}}(p_0, p_1, 0_{\mathbf{2}}) \equiv p_0$$
$$\mathsf{ind}_{\mathbf{2}}(p_0, p_1, 1_{\mathbf{2}}) \equiv p_1$$

hold.

Just as in the cases for the unit type and the empty type, the induction principle for the booleans can also be used to construct an ordinary function $\mathbf{2} \to A$, provided that we can construct two terms of type A. Indeed, by the induction principle for the booleans there is a function

$$\mathsf{ind}_{\mathbf{2}} : A \to (A \to A^{\mathbf{2}})$$

for any type A.

Example 4.4.2. Using the induction principle of $\mathbf{2}$ we can define all the operations of Boolean algebra. For example, the **boolean negation** operation $\mathsf{neg}_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ is defined by

$$\mathsf{neg}_{\mathbf{2}}(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}} \qquad\qquad \mathsf{neg}_{\mathbf{2}}(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}.$$

The **boolean conjunction** operation $- \wedge - : \mathbf{2} \to (\mathbf{2} \to \mathbf{2})$ is defined by

$$1_{\mathbf{2}} \wedge 1_{\mathbf{2}} :\equiv 1_{\mathbf{2}} \qquad\qquad 0_{\mathbf{2}} \wedge 1_{\mathbf{2}} :\equiv 0_{\mathbf{2}}$$
$$1_{\mathbf{2}} \wedge 0_{\mathbf{2}} :\equiv 0_{\mathbf{2}} \qquad\qquad 0_{\mathbf{2}} \wedge 0_{\mathbf{2}} :\equiv 0_{\mathbf{2}}.$$

The **boolean disjunction** operation $- \vee - : \mathbf{2} \to (\mathbf{2} \to \mathbf{2})$ is defined by

$$1_{\mathbf{2}} \vee 1_{\mathbf{2}} :\equiv 1_{\mathbf{2}} \qquad\qquad 0_{\mathbf{2}} \vee 1_{\mathbf{2}} :\equiv 1_{\mathbf{2}}$$
$$1_{\mathbf{2}} \vee 0_{\mathbf{2}} :\equiv 1_{\mathbf{2}} \qquad\qquad 0_{\mathbf{2}} \vee 0_{\mathbf{2}} :\equiv 0_{\mathbf{2}}.$$

We leave the definitions of some of the other boolean operations as Exercise 4.3. Note that the method of defining the boolean operations by the induction principle of $\mathbf{2}$ is not that different from defining them by truth tables.

Boolean logic is important, but it won't be very prominent in this course. The reason is simple: in type theory it is more natural to use the 'logic' of types that is provided by the inference rules.

4.5 Coproducts and the type of integers

Definition 4.5.1. Let A and B be types. We define the **coproduct** $A + B$ to be a type that comes equipped with

$$\mathsf{inl} : A \to A + B$$

$$\mathsf{inr} : B \to A + B$$

satisfying the induction principle that for any family of types $P(x)$ indexed by $x : A + B$, there is a term

$$\mathsf{ind}_+ : \left(\prod_{(x:A)} P(\mathsf{inl}(x))\right) \to \left(\prod_{(y:B)} P(\mathsf{inr}(y))\right) \to \prod_{(z:A+B)} P(z)$$

for which the computation rules

$$\begin{aligned} \mathsf{ind}_+(f, g, \mathsf{inl}(x)) &\equiv f(x) \\ \mathsf{inr}+(f, g, \mathsf{inr}(y)) &\equiv g(y) \end{aligned}$$

hold. Sometimes we write $[f, g]$ for $\mathsf{ind}_+(f, g)$.

The coproduct of two types is sometimes also called the **disjoint sum**. By the induction principle of coproducts it follows that we have a function

$$(A \to X) \to \big((B \to X) \to (A + B \to X)\big)$$

for any type X. Note that this special case of the induction principle of coproducts is very much like the elimination rule of disjunction in first order logic: if P, P', and Q are propositions, then we have

$$(P \to Q) \to \big((P' \to Q) \to (P \vee P' \to Q)\big).$$

Indeed, we can think of *propositions as types* and of terms as their constructive proofs. Under this interpretation of type theory the coproduct is indeed the disjunction.

An important example of a type that can be defined using coproducts is the type $\mathbb{Z}$ of integers.

Definition 4.5.2. We define the **integers** to be the type $\mathbb{Z} :\equiv \mathbb{N} + (\mathbf{1} + \mathbb{N})$. The type of integers comes equipped with inclusion functions of the positive and negative integers

$$\begin{aligned} \mathsf{in\text{-}pos} &:\equiv \mathsf{inr} \circ \mathsf{inr} \\ \mathsf{in\text{-}neg} &:\equiv \mathsf{inl}, \end{aligned}$$

which are both of type $\mathbb{N} \to \mathbb{Z}$, and the constants

$$\begin{aligned} -1_{\mathbb{Z}} &:\equiv \mathsf{in\text{-}neg}(0) \\ 0_{\mathbb{Z}} &:\equiv \mathsf{inr}(\mathsf{inl}(\star)) \\ 1_{\mathbb{Z}} &:\equiv \mathsf{in\text{-}pos}(0). \end{aligned}$$

In the following lemma we derive an induction principle for $\mathbb{Z}$, which can be used in many familiar constructions on $\mathbb{Z}$, such as in the definitions of addition and multiplication.

Lemma 4.5.3. *Consider a type family P over $\mathbb{Z}$. If we are given*

$$\begin{aligned} p_{-1} &: P(-1_{\mathbb{Z}}) \\ p_{-S} &: \prod_{(n:\mathbb{N})} P(\mathsf{in\text{-}neg}(n)) \to P(\mathsf{in\text{-}neg}(\mathsf{succ}_{\mathbb{N}}(n))) \\ p_0 &: P(0_{\mathbb{Z}}) \\ p_1 &: P(1_{\mathbb{Z}}) \\ p_S &: \prod_{(n:\mathbb{N})} P(\mathsf{in\text{-}pos}(n)) \to P(\mathsf{in\text{-}pos}(\mathsf{succ}_{\mathbb{N}}(n))), \end{aligned}$$

then we can construct a dependent function $f : \prod_{(k:\mathbb{Z})} P(k)$ *for which the following judgmental equalities hold:*

$$\begin{aligned}
f(-1_{\mathbb{Z}}) &\equiv p_{-1} \\
f(\mathsf{in\text{-}neg}(\mathsf{succ}_{\mathbb{N}}(n))) &\equiv p_{-S}(n, f(\mathsf{in\text{-}neg}(n))) \\
f(0_{\mathbb{Z}}) &\equiv p_0 \\
f(1_{\mathbb{Z}}) &\equiv p_1 \\
f(\mathsf{in\text{-}pos}(\mathsf{succ}_{\mathbb{N}}(n))) &\equiv p_S(n, f(\mathsf{in\text{-}pos}(n))).
\end{aligned}$$

Proof. Since $\mathbb{Z}$ is the coproduct of $\mathbb{N}$ and $\mathbf{1} + \mathbb{N}$, it suffices to define

$$\begin{aligned}
p_{inl} &: \prod_{(n:\mathbb{N})} P(\mathsf{inl}(n)) \\
p_{inr} &: \prod_{(t:\mathbf{1}+\mathbb{N})} P(\mathsf{inr}(t)).
\end{aligned}$$

Note that $\mathsf{in\text{-}neg} \equiv \mathsf{inl}$ and $-1_{\mathbb{Z}} \equiv \mathsf{in\text{-}neg}(0_{\mathbb{N}})$. In order to define p_{inl} we use induction on the natural numbers, so it suffices to define

$$\begin{aligned}
&p_{-1} : P(-1) \\
&p_{-S} : \prod_{(n:\mathbb{N})} P(\mathsf{in\text{-}neg}(n)) \to P(\mathsf{in\text{-}neg}(\mathsf{succ}_{\mathbb{N}}(n))).
\end{aligned}$$

Similarly, we proceed by coproduct induction, followed by induction on $\mathbf{1}$ in the left case and induction on $\mathbb{N}$ on the right case, in order to define p_{inr}. □

As an application we define the successor function on the integers.

Definition 4.5.4. We define the **successor function** on the integers $\mathsf{succ}_{\mathbb{Z}} : \mathbb{Z} \to \mathbb{Z}$ using the induction principle of Lemma 4.5.3, taking

$$\begin{aligned}
\mathsf{succ}_{\mathbb{Z}}(-1_{\mathbb{Z}}) &:\equiv 0_{\mathbb{N}} \\
\mathsf{succ}_{\mathbb{Z}}(\mathsf{in\text{-}neg}(\mathsf{succ}_{\mathbb{N}}(n))) &:\equiv \mathsf{in\text{-}neg}(n) \\
\mathsf{succ}_{\mathbb{Z}}(0_{\mathbb{Z}}) &:\equiv 1_{\mathbb{N}} \\
\mathsf{succ}_{\mathbb{Z}}(1_{\mathbb{Z}}) &:\equiv \mathsf{in\text{-}pos}(1_{\mathbb{N}}) \\
\mathsf{succ}_{\mathbb{Z}}(\mathsf{in\text{-}pos}(\mathsf{succ}_{\mathbb{N}}(n))) &:\equiv \mathsf{in\text{-}pos}(\mathsf{succ}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n))).
\end{aligned}$$

4.6 Dependent pair types

Given a type family B over A, we may consider pairs (a, b) of terms, where $a : A$ and $b : B(a)$. Note that the type of b depends on the first term in the pair, so we call such a pair a **dependent pair**.

The *dependent pair type* is an inductive type that is generated by the dependent pairs.

Definition 4.6.1. Consider a type family B over A. The **dependent pair type** (or Σ-type) is defined to be the inductive type $\sum_{(x:A)} B(x)$ equipped with a **pairing function**

$$(-,-) : \prod_{(x:A)} \Big(B(x) \to \textstyle\sum_{(y:A)} B(y)\Big).$$

The induction principle for $\sum_{(x:A)} B(x)$ asserts that for any family of types $P(p)$ indexed by $p : \sum_{(x:A)} B(x)$, there is a function

$$\mathsf{ind}_{\Sigma} : \Big(\prod_{(x:A)} \prod_{(y:B(x))} P(x, y)\Big) \to \Big(\prod_{(p:\sum_{(x:A)} B(x))} P(p)\Big).$$

satisfying the computation rule

$$\mathsf{ind}_\Sigma(f,(x,y)) \equiv f(x,y).$$

Sometimes we write $\lambda(x,y).\, f(x,y)$ for $\mathsf{ind}_\Sigma(\lambda x.\, \lambda y.\, f(x,y))$.

Definition 4.6.2. Given a type A and a type family B over A, the **first projection map**

$$\mathsf{pr}_1 : \left(\textstyle\sum_{(x:A)} B(x)\right) \to A$$

is defined by induction as

$$\mathsf{pr}_1 :\equiv \lambda(x,y).\, x.$$

The **second projection map** is a dependent function

$$\mathsf{pr}_2 : \textstyle\prod_{(p:\sum_{(x:A)} B(x))} B(\mathsf{pr}_1(p))$$

defined by induction as

$$\mathsf{pr}_2 :\equiv \lambda(x,y).\, y.$$

By the computation rule we have

$$\begin{aligned} \mathsf{pr}_1(x,y) &\equiv x \\ \mathsf{pr}_2(x,y) &\equiv y. \end{aligned}$$

4.7 Cartesian products

A special case of the Σ-type occurs when the B is a constant family over A, i.e., when B is just a type. In this case, the inductive type $\sum_{(x:A)} B(x)$ is generated by *ordinary* pairs (x,y) where $x : A$ and $y : B$. In other words, if B does not depend on A, then the type $\sum_{(x:A)} B$ is the *(cartesian) product* $A \times B$. The cartesian product is a very common special case of the dependent pair type, just as the type $A \to B$ of ordinary functions from $A \to B$ is a common special case of the dependent product. Therefore we provide its specification along with the induction principle for cartesian products.

Definition 4.7.1. Consider two types A and B. The **(cartesian) product** of A and B is defined as the inductive type $A \times B$ with constructor

$$(-,-) : A \to (B \to A \times B).$$

The induction principle for $A \times B$ asserts that for any type family P over $A \times B$, one has

$$\mathsf{ind}_\times : \left(\textstyle\prod_{(x:A)}\prod_{(y:B)} P(a,b)\right) \to \left(\textstyle\prod_{(p:A\times B)} P(p)\right)$$

satisfying the computation rule that

$$\mathsf{ind}_\times(f,(x,y)) \equiv f(x,y).$$

The projection maps are defined similarly to the projection maps of Σ-types. When one thinks of types as propositions, then $A \times B$ is interpreted as the conjunction of A and B.

Exercises

4.1 Write the rules for $\mathbf{1}$, $\emptyset$, $\mathbf{2}$, $A + B$, $\sum_{(x:A)} B(x)$, and $A \times B$. As usual, present the rules in four sets:

(i) A formation rule.

(ii) Introduction rules.

(iii) An elimination rule.

(iv) Computation rules.

4.2 Let P and Q be types. Use the fact that $\neg P$ is defined as the type $P \to \emptyset$ of functions from P to the empty type, to give type theoretic proofs of the following taugologies of constructive logic.

(a) $P \to \neg\neg P$
(b) $(P \to Q) \to (\neg\neg P \to \neg\neg Q)$
(c) $(P + \neg P) \to (\neg\neg P \to P)$
(d) $\neg\neg(P + \neg P)$
(e) $\neg\neg(\neg\neg P \to P)$
(f) $(P \to \neg\neg Q) \to (\neg\neg P \to \neg\neg Q)$
(g) $\neg\neg\neg P \to \neg P$
(h) $\neg\neg(P \to \neg\neg Q) \to (P \to \neg\neg Q)$
(i) $\neg\neg((\neg\neg P) \times (\neg\neg Q)) \to (\neg\neg P) \times (\neg\neg Q)$

4.3 Define the following operations of Boolean algebra:

exclusive disjunction	$p \oplus q$
implication	$p \Rightarrow q$
if and only if	$p \Leftrightarrow q$
Peirce's arrow (neither … nor)	$p \downarrow q$
Sheffer stroke (not both)	$p \mid q$.

Here p and q range over $\mathbf{2}$.

4.4 Define the predecessor function $\mathsf{pred}_{\mathbb{Z}} : \mathbb{Z} \to \mathbb{Z}$.

4.5 Define the group operations

$$\mathsf{add}_{\mathbb{Z}} : \mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z})$$
$$\mathsf{neg}_{\mathbb{Z}} : \mathbb{Z} \to \mathbb{Z},$$

and define the multiplication

$$\mathsf{mul}_{\mathbb{Z}} : \mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z}).$$

4.6 Construct a function $F : \mathbb{Z} \to \mathbb{Z}$ that extends the Fibonacci sequence to the negative integers

$$\ldots, 5, -3, 2, -1, 1, 0, 1, 1, 2, 3, 5, 8, 13, \ldots$$

in the expected way.

4.7 Show that $\mathbf{1} + \mathbf{1}$ satisfies the same induction principle as $\mathbf{2}$, i.e., define

$$t_0 : \mathbf{1} + \mathbf{1}$$
$$t_1 : \mathbf{1} + \mathbf{1},$$

and show that for any type family P over $\mathbf{1}+\mathbf{1}$ there is a function

$$\mathsf{ind}_{\mathbf{1}+\mathbf{1}} : P(t_0) \to \Big(P(t_1) \to \prod_{(t:\mathbf{1}+\mathbf{1})} P(t)\Big)$$

satisfying

$$\begin{aligned}\mathsf{ind}_{\mathbf{1}+\mathbf{1}}(p_0, p_1, t_0) &\equiv p_0 \\ \mathsf{ind}_{\mathbf{1}+\mathbf{1}}(p_0, p_1, t_1) &\equiv p_1.\end{aligned}$$

In other words, *type theory cannot distinguish between the types* $\mathbf{2}$ *and* $\mathbf{1}+\mathbf{1}$.

4.8 For any type A we can define the type $\mathsf{list}(A)$ of **lists** elements of A as the inductive type with constructors

$$\begin{aligned}\mathsf{nil} &: \mathsf{list}(A) \\ \mathsf{cons} &: A \to (\mathsf{list}(A) \to \mathsf{list}(A)).\end{aligned}$$

(a) Write down the induction principle and the computation rules for $\mathsf{list}(A)$.

(b) Let A and B be types, suppose that $b : B$, and consider a binary operation $\mu : A \to (B \to B)$. Define a function

$$\mathsf{fold\text{-}list}(\mu) : \mathsf{list}(A) \to B$$

that iterates the operation μ, starting with $\mathsf{fold\text{-}list}(\mu, \mathsf{nil}) :\equiv b$.

(c) Define a function $\mathsf{length\text{-}list} : \mathsf{list}(A) \to \mathbb{N}$.

(d) Define a function

$$\mathsf{sum\text{-}list} : \mathsf{list}(\mathbb{N}) \to \mathbb{N}$$

that adds all the elements in a list of natural numbers.

(e) Define a function

$$\mathsf{concat\text{-}list} : \mathsf{list}(A) \to (\mathsf{list}(A) \to \mathsf{list}(A))$$

that concatenates any two lists of elements in A.

(f) Define a function

$$\mathsf{flatten\text{-}list} : \mathsf{list}(\mathsf{list}(A)) \to \mathsf{list}(A)$$

that concatenates all the lists in a lists of lists in A.

(g) Define a function $\mathsf{reverse\text{-}list} : \mathsf{list}(A) \to \mathsf{list}(A)$ that reverses the order of the elements in any list.

5 Identity types

From the perspective of types as proof-relevant propositions, how should we think of *equality* in type theory? Given a type A, and two terms $x, y : A$, the equality $x = y$ should again be a type. Indeed, we want to *use* type theory to prove equalities. *Dependent* type theory provides us with a convenient setting for this: the equality type $x = y$ is dependent on $x, y : A$.

Then, if $x = y$ is to be a type, how should we think of the terms of $x = y$. A term $p : x = y$ witnesses that x and y are equal terms of type A. In other words $p : x = y$ is an *identification* of x and y. In a proof-relevant world, there might be many terms of type $x = y$. I.e., there might be many identifications of x and y. And, since $x = y$ is itself a type, we can form the type $p = q$ for any two identifications $p, q : x = y$. That is, since $x = y$ is a type, we may also use the type theory to prove things *about* identifications (for instance, that two given such identifications can

Table I.1: The homotopy interpretation

Type theory	*Homotopy theory*
Types	Spaces
Dependent types	Fibrations
Terms	Points
Dependent pair type	Total space
Identity type	Path fibration

themselves be identified), and we may use the type theory to perform constructions with them. As we will see shortly, we can give every type a groupoidal structure.

Clearly, the equality type should not just be any type dependent on $x, y : A$. Then how do we form the equality type, and what ways are there to use identifications in constructions in type theory? The answer to both these questions is that we will form the identity type as an *inductive* type, generated by just a reflexivity term providing an identification of x to itself. The induction principle then provides us with a way of performing constructions with identifications, such as concatenating them, inverting them, and so on. Thus, the identity type is equipped with a reflexivity term, and further possesses the structure that are generated by its induction principle and by the type theory. This inductive construction of the identity type is elegant, beautifully simple, but far from trivial!

The situation where two terms can be identified in possibly more than one way is analogous to the situation in *homotopy theory*, where two points of a space can be connected by possibly more than one *path*. Indeed, for any two points x, y in a space, there is a *space of paths* from x to y. Moreover, between any two paths from x to y there is a space of *homotopies* between them, and so on. This leads to the homotopy interpretation of type theory, outlined in Table I.1. The connection between homotopy theory and type theory been made precise by the construction of homotopical models of type theory, and it has led to the fruitful research area of *synthetic homotopy theory*, the subfield of *homotopy type theory* that is the topic of this course.

5.1 The inductive definition of identity types

Definition 5.1.1. Consider a type A and let $a : A$. Then we define the **identity type** of A at a as an inductive family of types $a =_A x$ indexed by $x : A$, of which the constructor is

$$\mathsf{refl}_a : a =_A a.$$

The induction principle of the identity type postulates that for any family of types $P(x, p)$ indexed by $x : A$ and $p : a =_A x$, there is a function

$$\mathsf{path\text{-}ind}_a : P(a, \mathsf{refl}_a) \to \prod_{(x:A)} \prod_{(p:a=_A x)} P(x, p)$$

that satisfies $\mathsf{path\text{-}ind}_a(p, a, \mathsf{refl}_a) \equiv p$.

A term of type $a =_A x$ is also called an **identification** of a with x, and sometimes it is called a **path** from a to x. The induction principle for identity types is sometimes called **identification elimination** or **path induction**. We also write Id_A for the identity type on A, and often we write $a = x$ for the type of identifications of a with x, omitting reference to the ambient type A.

Remark 5.1.2. We see that the identity type is not just an inductive type, like the inductive types $\mathbb{N}$, $\emptyset$, and $\mathbf{1}$ for example, but it is and inductive *family* of types. Even though we have a type

$a =_A x$ for any $x : A$, the constructor only provides a term $\mathsf{refl}_a : a =_A a$, identifying a with itself. The induction principle then asserts that in order to prove something about all identifications of a with some $x : A$, it suffices to prove this assertion about refl_a only. We will see in the next sections that this induction principle is strong enough to derive many familiar facts about equality, namely that it is a symmetric and transitive relation, and that all functions preserve equality.

Remark 5.1.3. Since the identity types require getting used to, we provide the formal rules for identity types. The identity type is formed by the formation rule:

$$\frac{\Gamma \vdash a : A}{\Gamma, x : A \vdash a =_A x \text{ type}}$$

The constructor of the identity type is then given by the introduction rule:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}$$

The induction principle is now given by the elimination rule:

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x : A, p : a =_A x \vdash P(x, p) \text{ type}}{\Gamma \vdash \mathsf{path\text{-}ind}_a : P(a, \mathsf{refl}_a) \to \prod_{(x:A)} \prod_{(p:a=_A x)} P(x, p)}$$

And finally the computation rule is:

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x : A, p : a =_A x \vdash P(x, p) \text{ type}}{\Gamma \vdash \mathsf{path\text{-}ind}_a(p, a, \mathsf{refl}_a) \equiv p : P(a, \mathsf{refl}_a)}$$

Remark 5.1.4. One might wonder whether it is also possible to form the identity type at a *variable* of type A, rather than at a term. This is certainly possible: since we can form the identity type in *any* context, we can form the identity type at a variable $x : A$ as follows:

$$\frac{\Gamma, x : A \vdash x : A}{\Gamma, x : A, y : A \vdash x =_A y \text{ type}}$$

In this way we obtain the 'binary' identity type. Its constructor is then also indexed by $x : A$. We have the following introduction rule

$$\frac{\Gamma, x : A \vdash x : A}{\Gamma, x : A \vdash \mathsf{refl}_x : x =_A x}$$

and similarly we have elimination and computation rules.

5.2 The groupoidal structure of types

We show that identifications can be *concatenated* and *inverted*, which corresponds to the transitivity and symmetry of the identity type.

Definition 5.2.1. Let A be a type. We define the **concatenation** operation

$$\mathsf{concat} : \prod_{(x,y,z:A)} (x = y) \to (y = z) \to (x = z).$$

We will write $p \cdot q$ for $\mathsf{concat}(p, q)$.

Construction. We construct the concatenation operation by path induction. It suffices to construct

$$\mathsf{concat}(\mathsf{refl}_x) : \prod_{(z:A)} (x = z) \to (x = z).$$

Here we take $\mathsf{concat}(\mathsf{refl}_x)_z \equiv \mathsf{id}_{(x=z)}$. Explicitly, the term we have constructed is

$$\lambda x.\, \mathsf{path\text{-}ind}_x(\lambda z.\, \mathsf{id}_{(x=z)}) : \prod_{(x,y:A)} (x = y) \to \prod_{(z:A)} (y = z) \to (x = z).$$

To obtain a term of the asserted type we need to swap the order of the arguments $p : x = y$ and $z : A$, using Exercise 2.5. □

Definition 5.2.2. Let A be a type. We define the **inverse operation**

$$\mathsf{inv} : \prod_{(x,y:A)} (x = y) \to (y = x).$$

Most of the time we will write p^{-1} for $\mathsf{inv}(p)$.

Construction. We construct the inverse operation by path induction. It suffices to construct

$$\mathsf{inv}(\mathsf{refl}_x) : x = x,$$

for any $x : A$. Here we take $\mathsf{inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_x$. □

The next question is whether the concatenation and inverting operations on paths behave as expected. More concretely, is path concatenation associative, does it satisfy the unit laws, and is the inverse of a path indeed a two-sided inverse?

For example, in the case of associativity we are asking to compare the paths

$$(p \cdot q) \cdot r \qquad \text{and} \qquad p \cdot (q \cdot r)$$

for any $p : x = y$, $q : y = z$, and $r : z = w$ in a type A. The computation rules of path induction are not strong enough to conclude that $(p \cdot q) \cdot r$ and $p \cdot (q \cdot r)$ are judgmentally equal. However, both $(p \cdot q) \cdot r$ and $p \cdot (q \cdot r)$ are terms of the same type: they are identifications of type $x = w$. Since the identity type is a type like any other, we can ask whether there is an *identification*

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

This is a very useful idea: while it is often impossible to show that two terms of the same type are judgmentally equal, it may be the case that those two terms can be *identified*. Indeed, we identify two terms by constructing a term of the identity type, and we can use all the type theory at our disposal in order to construct such a term. In this way we can show, for example, that addition on the natural numbers or on the integers is associative and satisfies the unit laws. And indeed, here we will show that path concatenation is associative and satisfies the unit laws.

Definition 5.2.3. Let A be a type and consider three consecutive paths

$$x \overset{p}{=\!=} y \overset{q}{=\!=} z \overset{r}{=\!=} w$$

in A. We define the **associator**

$$\mathsf{assoc}(p,q,r) : (p \cdot q) \cdot r = p \cdot (q \cdot r).$$

Construction. By path induction it suffices to show that

$$\prod_{(z:A)} \prod_{(q:x=z)} \prod_{(w:A)} \prod_{(r:z=w)} (\mathsf{refl}_x \cdot q) \cdot r = \mathsf{refl}_x \cdot (q \cdot r).$$

Let $q : x = z$ and $r : z = w$. Note that by the computation rule of the path induction principle we have a judgmental equality $\mathsf{refl}_x \cdot q \equiv q$. Therefore we conclude that

$$(\mathsf{refl}_x \cdot q) \cdot r \equiv q \cdot r.$$

Similarly we have a judgmental equality $\mathsf{refl}_x \cdot (q \cdot r) \equiv q \cdot r$. Thus we see that the left-hand side and the right-hand side in

$$(\mathsf{refl}_x \cdot q) \cdot r = \mathsf{refl}_x \cdot (q \cdot r)$$

are judgmentally equal, so we can simply define $\mathsf{assoc}(\mathsf{refl}_x, q, r) :\equiv \mathsf{refl}_{q \cdot r}$. □

Definition 5.2.4. Let A be a type. We define the left and right **unit law operations**, which assigns to each $p : x = y$ the terms

$$\mathsf{left\text{-}unit}(p) : \mathsf{refl}_x \cdot p = p$$
$$\mathsf{right\text{-}unit}(p) : p \cdot \mathsf{refl}_y = p,$$

respectively.

Construction. By identification elimination it suffices to construct

$$\mathsf{left\text{-}unit}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x$$
$$\mathsf{right\text{-}unit}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x.$$

In both cases we take $\mathsf{refl}_{\mathsf{refl}_x}$. □

Definition 5.2.5. Let A be a type. We define left and right **inverse law operations**

$$\mathsf{left\text{-}inv}(p) : p^{-1} \cdot p = \mathsf{refl}_y$$
$$\mathsf{right\text{-}inv}(p) : p \cdot p^{-1} = \mathsf{refl}_x.$$

Construction. By identification elimination it suffices to construct

$$\mathsf{left\text{-}inv}(\mathsf{refl}_x) : \mathsf{refl}_x^{-1} \cdot \mathsf{refl}_x = \mathsf{refl}_x$$
$$\mathsf{right\text{-}inv}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x^{-1} = \mathsf{refl}_x.$$

Using the computation rules we see that

$$\mathsf{refl}_x^{-1} \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x,$$

so we define $\mathsf{left\text{-}inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}$. Similarly it follows from the computation rules that

$$\mathsf{refl}_x \cdot \mathsf{refl}_x^{-1} \equiv \mathsf{refl}_x^{-1} \equiv \mathsf{refl}_x$$

so we again define $\mathsf{right\text{-}inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}$. □

Remark 5.2.6. We have seen that the associator, the unit laws, and the inverse laws, are all proven by constructing an identification of identifications. And indeed, there is nothing that would stop us from considering identifications of those identifications of identifications. We can go up as far as we like in the *tower of identity types*, which is obtained by iteratively taking identity types.

The iterated identity types give types in homotopy type theory a very intricate structure. One important way of studying this structure is via the homotopy groups of types, a subject that we will gradually be working towards.

5.3 The action on paths of functions

Using the induction principle of the identity type we can show that every function preserves identifications. In other words, every function sends identified terms to identified terms. Note that this is a form of continuity for functions in type theory: if there is a path that identifies two points x and y of a type A, then there also is a path that identifies the values $f(x)$ and $f(y)$ in the codomain of f.

Definition 5.3.1. Let $f : A \to B$ be a map. We define the **action on paths** of f as an operation

$$\mathsf{ap}_f : \prod_{(x,y:A)} (x = y) \to (f(x) = f(y)).$$

Moreover, there are operations

$$\begin{aligned} \mathsf{ap\text{-}id}_A &: \prod_{(x,y:A)} \prod_{(p:x=y)} p = \mathsf{ap}_{\mathsf{id}_A}(p) \\ \mathsf{ap\text{-}comp}(f,g) &: \prod_{(x,y:A)} \prod_{(p:x=y)} \mathsf{ap}_g(\mathsf{ap}_f(p)) = \mathsf{ap}_{g \circ f}(p). \end{aligned}$$

Construction. First we define ap_f by identity elimination, taking

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

Next, we construct $\mathsf{ap\text{-}id}_A$ by identity elimination, taking

$$\mathsf{ap\text{-}id}_A(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}.$$

Finally, we construct $\mathsf{ap\text{-}comp}(f,g)$ by identity elimination, taking

$$\mathsf{ap\text{-}comp}(f,g,\mathsf{refl}_x) :\equiv \mathsf{refl}_{g(f(x))}. \qquad \square$$

Definition 5.3.2. Let $f : A \to B$ be a map. Then there are identifications

$$\mathsf{ap\text{-}refl}(f,x) : \mathsf{ap}_f(\mathsf{refl}_x) = \mathsf{refl}_f(x)$$

$$\mathsf{ap\text{-}inv}(f,p) : \mathsf{ap}_f(p^{-1}) = \mathsf{ap}_f(p)^{-1}$$

$$\mathsf{ap\text{-}concat}(f,p,q) : \mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q)$$

for every $p : x = y$ and $q : x = y$.

Construction. To construct $\mathsf{ap\text{-}refl}(f,x)$ we simply observe that $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_f(x)$, so we take

$$\mathsf{ap\text{-}refl}(f,x) :\equiv \mathsf{refl}_{\mathsf{refl}_{f(x)}}.$$

We construct $\mathsf{ap\text{-}inv}(f,p)$ by identification elimination on p, taking

$$\mathsf{ap\text{-}inv}(f,\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{ap}_f(\mathsf{refl}_x)}.$$

Finally we construct $\mathsf{ap\text{-}concat}(f,p,q)$ by identification elimination on p, taking

$$\mathsf{ap\text{-}concat}(f,\mathsf{refl}_x,q) :\equiv \mathsf{refl}_{\mathsf{ap}_f(q)}. \qquad \square$$

5.4 Transport

Dependent types also come with an action on paths: the *transport* functions. Given an identification $p : x = y$ in the base type A, we can transport any term $b : B(x)$ to the fiber $B(y)$. The transport functions have many applications, which we will encounter throughout this course.

Definition 5.4.1. Let A be a type, and let B be a type family over A. We will construct a **transport** operation

$$\mathsf{tr}_B : \prod_{(x,y:A)}(x = y) \to (B(x) \to B(y)).$$

Construction. We construct $\mathsf{tr}_B(p)$ by induction on $p : x =_A y$, taking

$$\mathsf{tr}_B(\mathsf{refl}_x) :\equiv \mathsf{id}_{B(x)}.$$

□

Thus we see that type theory cannot distinguish between identified terms x and y, because for any type family B over A one gets a term of $B(y)$ as soon as $B(x)$ has a term.

As an application of the transport function we construct the *dependent* action on paths of a dependent function $f : \prod_{(x:A)} B(x)$. Note that for such a dependent function f, and an identification $p : x =_A y$, it does not make sense to directly compare $f(x)$ and $f(y)$, since the type of $f(x)$ is $B(x)$ whereas the type of $f(y)$ is $B(y)$, which might not be exactly the same type. However, we can first *transport* $f(x)$ along p, so that we obtain the term $\mathsf{tr}_B(p, f(x))$ which is of type $B(y)$. Now we can ask whether it is the case that $\mathsf{tr}_B(p, f(x)) = f(y)$. The dependent action on paths of f establishes this identification.

Definition 5.4.2. Given a dependent function $f : \prod_{(a:A)} B(a)$ and a path $p : x = y$ in A, we construct a path

$$\mathsf{apd}_f(p) : \mathsf{tr}_B(p, f(x)) = f(y).$$

Construction. The path $\mathsf{apd}_f(p)$ is constructed by path induction on p. Thus, it suffices to construct a path

$$\mathsf{apd}_f(\mathsf{refl}_x) : \mathsf{tr}_B(\mathsf{refl}_x, f(x)) = f(x).$$

Since transporting along refl_x is the identity function on $B(x)$, we simply take $\mathsf{apd}_f(\mathsf{refl}_x) :\equiv \mathsf{refl}_{f(x)}$. □

Exercises

5.1 (a) State Goldbach's Conjecture in type theory.
 (b) State the Twin Prime Conjecture in type theory.

5.2 Show that the operation inverting paths distributes over the concatenation operation, i.e., construct an identification

$$\mathsf{distributive\text{-}inv\text{-}concat}(p, q) : (p \cdot q)^{-1} = q^{-1} \cdot p^{-1}.$$

for any $p : x = y$ and $q : y = z$.

5.3 For any $p : x = y$, $q : y = z$, and $r : x = z$, construct maps

$$\mathsf{inv\text{-}con}(p, q, r) : (p \cdot q = r) \to (q = p^{-1} \cdot r)$$

$$\mathsf{con\text{-}inv}(p, q, r) : (p \cdot q = r) \to (p = r \cdot q^{-1}).$$

5.4 Let B be a type family over A, and consider a path $p : x = x'$ in A. Construct for any $y : B(x)$ a path

$$\mathsf{lift}_B(p, y) : (x, y) = (x', \mathsf{tr}_B(p, y)).$$

In other words, a path in the *base type A lifts* to a path in the total space $\sum_{(x:A)} B(x)$ for every term over the domain, analogous to the path lifting property for fibrations in homotopy theory.

5.5 In this exercise we show that the operations of addition and multiplication on the natural numbers satisfy the laws of a commutative **semi-ring**.

(a) Show that addition satisfies the following laws:

$$\begin{aligned} m + 0 &= m & \qquad m + \mathsf{succ}_{\mathbb{N}}(n) &= \mathsf{succ}_{\mathbb{N}}(m+n) \\ 0 + m &= m & \qquad \mathsf{succ}_{\mathbb{N}}(m) + n &= \mathsf{succ}_{\mathbb{N}}(m+n). \end{aligned}$$

(b) Show that addition is associative and commutative, i.e., show that we have identifications

$$\begin{aligned} (m+n)+k &= m+(n+k) \\ m+n &= n+m. \end{aligned}$$

(c) Show that multiplication satisfies the following laws:

$$\begin{aligned} m \cdot 0 &= 0 & \qquad m \cdot 1 &= m & \qquad m \cdot \mathsf{succ}_{\mathbb{N}}(n) &= m + m \cdot n \\ 0 \cdot m &= 0 & \qquad 1 \cdot m &= m & \qquad \mathsf{succ}_{\mathbb{N}}(m) \cdot n &= m \cdot n + n. \end{aligned}$$

(d) Show that multiplication on $\mathbb{N}$ is commutative:

$$m \cdot n = n \cdot m.$$

(e) Show that multiplication on $\mathbb{N}$ distributes over addition from the left and from the right, i.e., show that we have identifications

$$\begin{aligned} m \cdot (n+k) &= m \cdot n + m \cdot k \\ (m+n) \cdot k &= m \cdot k + n \cdot k. \end{aligned}$$

(f) Show that multiplication on $\mathbb{N}$ is associative:

$$(m \cdot n) \cdot k = m \cdot (n \cdot k).$$

5.6 Consider four consecutive identifications

$$a \overset{p}{=\!=} b \overset{q}{=\!=} c \overset{r}{=\!=} d \overset{s}{=\!=} e$$

in a type A. In this exercise we will show that the **Mac Lane pentagon** for identifications commutes.

(a) Construct the five identifications $\alpha_1, \ldots, \alpha_5$ in the pentagon

$$\begin{tikzcd}[column sep=small] & ((p \cdot q) \cdot r) \cdot s \arrow[rr, equal, "\alpha_4"] \arrow[dl, equal, "\alpha_1"'] & & (p \cdot q) \cdot (r \cdot s) \arrow[dr, equal, "\alpha_5"] & \\ (p \cdot (q \cdot r)) \cdot s \arrow[drr, equal, "\alpha_2"'] & & & & p \cdot (q \cdot (r \cdot s)), \\ & & p \cdot ((q \cdot r) \cdot s) \arrow[urr, equal, "\alpha_3"'] & & \end{tikzcd}$$

where α_1, α_2, and α_3 run counter-clockwise, and α_4 and α_5 run clockwise.

(b) Show that
$$(\alpha_1 \cdot \alpha_2) \cdot \alpha_3 = \alpha_4 \cdot \alpha_5.$$

6 Universes

To complete our specification of dependent type theory, we introduce type theoretic *universes*. Universes are types that consist of types. In other words, a universe is a type $\mathcal{U}$ that comes equipped with a type family $\mathcal{T}$ over $\mathcal{U}$, and for any $X : \mathcal{U}$ we think of X as an *encoding* of the type $\mathcal{T}(X)$. We call this type family the *universal type family*.

There are several reasons to equip type theory with universes. One reason is that it enables us to define new type families over inductive types, using their induction principle. For example, since the universe is itself a type, we can use the induction principle of $\mathbf{2}$ to obtain a map $P : \mathbf{2} \to \mathcal{U}$ from any two terms $X_0, X_1 : \mathcal{U}$. Then we obtain a type family over $\mathbf{2}$ by substituting P into the universal type family:
$$x : \mathbf{2} \vdash \mathcal{T}(P(x)) \text{ type}$$
satisfying $\mathcal{T}(P(0_{\mathbf{2}})) \equiv \mathcal{T}(X_0)$ and $\mathcal{T}(P(1_{\mathbf{2}})) \equiv \mathcal{T}(X_1)$.

We use this way of defining type families to define many familiar relations over $\mathbb{N}$, such as $\leq$ and $<$. We also introduce a relation called *observational equality* $\mathsf{Eq}_{\mathbb{N}}$ on $\mathbb{N}$, which we can think of as equality of $\mathbb{N}$. This relation is reflexive, symmetric, and transitive, and moreover it is the least reflexive relation. Furthermore, one of the most important aspects of observational equality $\mathsf{Eq}_{\mathbb{N}}$ on $\mathbb{N}$ is that $\mathsf{Eq}_{\mathbb{N}}(m, n)$ is a type for every $m, n : \mathbb{N}$, unlike judgmental equality. Therefore we can use type theory to reason about observational equality on $\mathbb{N}$. Indeed, in the exercises we show that some very elementary mathematics can already be done at this early stage in our development of type theory.

A second reason to introduce universes is that it allows us to define many types of types equipped with structure. One of the most important examples is the type of groups, which is the type of types equipped with the group operations satisfying the group laws, and for which the underlying type is a set. We won't discuss the condition for a type to be a set until §10, so the definition of groups in type theory will be given much later. Therefore we illustrate this use of the universe by giving simpler examples: pointed types, graphs, and reflexive graphs.

One of the aspects that make universes useful is that they are postulated to be closed under all the type constructors. For example, if we are given $X : \mathcal{U}$ and $P : \mathcal{T}(X) \to \mathcal{U}$, then the universe is equipped with a term
$$\check{\Sigma}(X, P) : \mathcal{U}$$
satisfying the judgmental equality $\mathcal{T}(\check{\Sigma}(X, P) \equiv \sum_{(x:\mathcal{T}(X))} \mathcal{T}(P(x))$. We will similarly assume that any universe is closed under Π-types and the other ways of forming types. However, there is an important restriction: it would be inconsistent to assume that the universe is contained in itself. One way of thinking about this is that universes are types of *small* types, and it cannot be the case that the universe is small with respect to itself. We address this problem by assuming that there are many universes: enough universes so that any type family can be obtained by substituting into the universal type family of some universe.

6.1 Specification of type theoretic universes

In the following definition we already state that universes are closed under identity types. Identity types will be introduced in §5.

Definition 6.1.1. A **universe** in type theory is a closed type $\mathcal{U}$ equipped with a type family $\mathcal{T}$ over $\mathcal{U}$ called the **universal family**, equipped with the following structure:

(i) $\mathcal{U}$ is closed under Π, in the sense that it comes equipped with a function

$$\check{\Pi} : \prod_{(X:\mathcal{U})}(\mathcal{T}(X) \to \mathcal{U}) \to \mathcal{U}$$

for which the judgmental equality

$$\mathcal{T}\big(\check{\Pi}(X, P)\big) \equiv \prod_{(x:\mathcal{T}(X))} \mathcal{T}(P(x)).$$

holds, for every $X : \mathcal{U}$ and $P : \mathcal{T}(X) \to \mathcal{U}$.

(ii) $\mathcal{U}$ is closed under Σ in the sense that it comes equipped with a function

$$\check{\Sigma} : \prod_{(X:\mathcal{U})}(\mathcal{T}(X) \to \mathcal{U}) \to \mathcal{U}$$

for which the judgmental equality

$$\mathcal{T}\big(\check{\Sigma}(X, P)\big) \equiv \sum_{(x:\mathcal{T}(X))} \mathcal{T}(P(x))$$

holds, for every $X : \mathcal{U}$ and $P : \mathcal{T}(X) \to \mathcal{U}$.

(iii) $\mathcal{U}$ is closed under identity types, in the sense that it comes equipped with a function

$$\check{\mathrm{I}} : \prod_{(X:\mathcal{U})} \mathcal{T}(X) \to (\mathcal{T}(X) \to \mathcal{U})$$

for which the judgmental equality

$$\mathcal{T}\big(\check{\mathrm{I}}(X, x, y)\big) \equiv (x = y)$$

holds, for every $X : \mathcal{U}$ and $x, y : \mathcal{T}(X)$.

(iv) $\mathcal{U}$ is closed under coproducts, in the sense that it comes equipped with a function

$$\check{+} : \mathcal{U} \to (\mathcal{U} \to \mathcal{U})$$

that satisfies $\mathcal{T}\big(X \check{+} Y\big) \equiv \mathcal{T}(X) + \mathcal{T}(Y)$.

(v) $\mathcal{U}$ contains terms $\check{\emptyset}, \check{\mathbf{1}}, \check{\mathbb{N}} : \mathcal{U}$ that satisfy the judgmental equalities

$$\mathcal{T}(\check{\emptyset}) \equiv \emptyset$$
$$\mathcal{T}(\check{\mathbf{1}}) \equiv \mathbf{1}$$
$$\mathcal{T}(\check{\mathbb{N}}) \equiv \mathbb{N}.$$

Given a universe $\mathcal{U}$, we say that a type A in context Γ is **small** with respect to $\mathcal{U}$ if it occurs in the universe, i.e., if it comes equipped with a term $\check{A} : \mathcal{U}$ in context Γ, for which the judgment

$$\Gamma \vdash \mathcal{T}(\check{A}) \equiv A \text{ type}$$

holds. If A is small with respect to $\mathcal{U}$, we usually write simply A for $\check{A}$ and also A for $\mathcal{T}(\check{A})$. In other words, by $A : \mathcal{U}$ we mean that A is a small type.

Remark 6.1.2. Since ordinary function types are defined as a special case of dependent function types, we don't have to assume that universes are closed under ordinary function types. Similarly, it follows from the assumption that universes are closed under dependent pair types that universes are closed under cartesian product types.

6.2 Assuming enough universes

Most of the time we will get by with assuming one universe $\mathcal{U}$, and indeed we recommend on a first reading of this text to simply assume that there is one universe $\mathcal{U}$. However, sometimes we might need a second universe $\mathcal{V}$ that contains $\mathcal{U}$ as well as all the types in $\mathcal{U}$. In such situations we cannot get by with a single universe, because the assumption that $\mathcal{U}$ is a term of itself would lead to inconsistencies like the Russel's paradox.

Russel's paradox is the famous argument that there cannot be a set of all sets. If there were such a set S, then we could consider Russel's subset

$$R := \{x \in S \mid x \notin x\}.$$

Russell then observed that $R \in R$ if and only if $R \notin R$, so we reach a contradiction. A variant of this argument reaches a similar contradiction when we assume that $\mathcal{U}$ is a universe that contains a term $\check{\mathcal{U}} : \mathcal{U}$ such that $\mathcal{T}(\check{\mathcal{U}}) \equiv \mathcal{U}$. In order to avoid such paradoxes, Russell and Whitehead formulated the *ramified theory of types* in their book *Principia Mathematica*. The ramified theory of types is a precursor of Martin Löf's type theory that we are studying in this course.

Even though the universe is not a term of itself, it is still convenient if every type, including any universe, is small with respect to *some* universe. Therefore we will assume that there are sufficiently many universes: we will assume that for every finite list of types

$$\begin{gathered}\Gamma_1 \vdash A_1 \text{ type} \\ \vdots \\ \Gamma_n \vdash A_n \text{ type},\end{gathered}$$

there is a universe $\mathcal{U}$ that contains each A_i in the sense that $\mathcal{U}$ comes equipped with a term

$$\Gamma_i \vdash \check{A}_i : \mathcal{U}$$

for which the judgment

$$\Gamma_i \vdash \mathcal{T}(\check{A}_i) \equiv A_i \text{ type}$$

holds. With this assumption it will rarely be necessary to work with more than one universe at the same time.

Remark 6.2.1. Using the assumption that for any finite list of types in context there is a universe that contains those types, we obtain many specific universes:

(i) There is a *base universe* $\mathcal{U}_0$ that we obtain using the empty list of types in context. This is a universe, but it isn't specified to contain any further types.

(ii) Given a finite list

$$\begin{gathered}\Gamma_1 \vdash A_1 \text{ type} \\ \vdots \\ \Gamma_n \vdash A_n \text{ type},\end{gathered}$$

of types in context, and a universe $\mathcal{U}$ that contains them, there is a universe $\mathcal{U}^+$ that contains all the types in $\mathcal{U}$ as well as $\mathcal{U}$. More precisely, it is specified by the finite list

$$\vdash \mathcal{U} \text{ type}$$
$$X : \mathcal{U} \vdash \mathcal{T}(X) \text{ type.}$$

The universe $\mathcal{U}^+$ therefore contains the type $\mathcal{U}$ as well as every type in $\mathcal{U}$, in the following sense

$$\begin{array}{cc} \vdash \check{\mathcal{U}} : \mathcal{U}^+ & \vdash \mathcal{T}^+(\check{\mathcal{U}}) \equiv \mathcal{U} \text{ type} \\ X : \mathcal{U} \vdash \check{\mathcal{T}}(X) : \mathcal{U}^+ & X : \mathcal{U} \vdash \mathcal{T}^+(\check{\mathcal{T}}(X)) \equiv \mathcal{T}(X) \text{ type.} \end{array}$$

In particular, we obtain a function $i : \mathcal{U} \to \mathcal{U}^+$ that includes the $\mathcal{U}$-small types into $\mathcal{U}^+$.

Note that since the universe $\mathcal{U}^+$ contains all the types in $\mathcal{U}$, it also contains the types $A_1, \ldots, A_n$. To see this, we derive that there is a code for A_i in $\mathcal{U}^+$.

$$\frac{\Gamma_i \vdash \check{A}_i : \mathcal{U} \qquad \dfrac{X : \mathcal{U} \vdash \check{\mathcal{T}}(X) : \mathcal{U}^+}{\Gamma_i, X : \mathcal{U} \vdash \check{\mathcal{T}}(X) : \mathcal{U}^+}}{\Gamma_i \vdash \check{\mathcal{T}}(\check{A}_i) : \mathcal{U}^+}$$

We leave it as an exercise to derive the judgmental equality

$$\mathcal{T}^+(\check{\mathcal{T}}(\check{A}_i)) \equiv A_i.$$

(iii) Given two finite lists

$$\begin{array}{cc} \Gamma_1 \vdash A_1 \text{ type} & \Delta_1 \vdash B_1 \text{ type} \\ \vdots & \vdots \\ \Gamma_n \vdash A_n \text{ type} & \Delta_m \vdash B_m \text{ type} \end{array}$$

of types in context, and two universes $\mathcal{U}$ and $\mathcal{V}$ that contain $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$ respectively, there is a universe $\mathcal{U} \sqcup \mathcal{V}$ that contains the types of both $\mathcal{U}$ and $\mathcal{V}$. The universe $\mathcal{U} \sqcup \mathcal{V}$ is specified by the finite list

$$X : \mathcal{U} \vdash \mathcal{T}_{\mathcal{U}}(X) \text{ type}$$
$$Y : \mathcal{V} \vdash \mathcal{T}_{\mathcal{V}}(Y) \text{ type.}$$

With an argument similar to the previous construction of a universe, we see that the universe $\mathcal{U} \sqcup \mathcal{V}$ contains the types $A_1, \ldots, A_n$ as well as the types $B_1, \ldots, B_m$.

Note that we could also directly obtain a universe $\mathcal{W}$ that contains the types $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$. However, this universe might not contain all the types in $\mathcal{U}$ or all the types in $\mathcal{V}$.

Since we don't postulate any relations between the universes, there are indeed very few of them. For example, the base universe $\mathcal{U}_0$ might contain many more types than it is postulated to contain. Nevertheless, there are some relations between the universes. For instance, there is a function $\mathcal{U} \to \mathcal{U}^+$, since we can simply derive

$$\frac{X : \mathcal{U} \vdash \check{\mathcal{T}}(X) : \mathcal{U}^+}{\vdash \lambda X.\, \check{\mathcal{T}}(X) : \mathcal{U} \to \mathcal{U}^+}$$

Similarly, there are functions $\mathcal{U} \to \mathcal{U} \sqcup \mathcal{V}$ and $\mathcal{V} \to \mathcal{U} \sqcup \mathcal{V}$ for any two universes $\mathcal{U}$ and $\mathcal{V}$.

6.3 Pointed types

Definition 6.3.1. A **pointed type** is a pair (A, a) consisting of a type A and a term $a : A$. The type of all pointed types in a universe $\mathcal{U}$ is defined to be

$$\mathcal{U}_* :\equiv \textstyle\sum_{(X:\mathcal{U})} X.$$

Definition 6.3.2. Consider two pointed types (A, a) and (B, b). A **pointed map** from (A, a) to (B, b) is a pair (f, p) consisting of a function $f : A \to B$ and an identification $p : f(a) = b$. We write

$$A \to_* B :\equiv \textstyle\sum_{(f:A \to B)} f(a) = b$$

for the type of all pointed maps from (A, a) to (B, b), leaving the base point implicit.

Since we have a type $\mathcal{U}_*$ of *all* pointed types in a universe $\mathcal{U}$, we can start defining operations on $\mathcal{U}_*$. An important example of such an operation is to take the loop space of a pointed type.

Definition 6.3.3. We define the **loop space** operation $\Omega : \mathcal{U}_* \to \mathcal{U}_*$

$$\Omega(A, a) :\equiv \big((a = a), \mathsf{refl}_a\big).$$

We can even go further and define the *iterated loop space* of a pointed type. Note that this definition could not be given in type theory if we didn't have universes.

Definition 6.3.4. Given a pointed type (A, a) and a natural number n, we define the n-th loop space $\Omega^n(A, a)$ by induction on $n : \mathbb{N}$, taking

$$\begin{aligned} \Omega^0(A, a) &:\equiv (A, a) \\ \Omega^{n+1}(A, a) &:\equiv \Omega(\Omega^n(A, a)). \end{aligned}$$

6.4 Families and relations on the natural numbers

As we have already seen in the case of the iterated loop space, we can use the universe to define a type family over $\mathbb{N}$ by induction on $\mathbb{N}$. For example, we can define the finite types in this way.

Definition 6.4.1. We define the type family $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$ of finite types by induction on $\mathbb{N}$, taking

$$\begin{aligned} \mathsf{Fin}(0_{\mathbb{N}}) &:\equiv \emptyset \\ \mathsf{Fin}(\mathsf{succ}_{\mathbb{N}}(n)) &:\equiv \mathsf{Fin}(n) + \mathbf{1} \end{aligned}$$

Similarly, we can define many relations on the natural numbers using a universe. We give here the example of *observational equality* on $\mathbb{N}$. This inductively defined equivalence relation is very important, as it can be used to show that equality on the natural numbers is *decidable*, i.e., there is a program that decides for any two natural numbers m and n whether they are equal or not.

Definition 6.4.2. We define the **observational equality** on $\mathbb{N}$ as binary relation $\mathsf{Eq}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ satisfying

$$\begin{aligned} \mathsf{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) &\equiv \mathbf{1} & \mathsf{Eq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n), 0_{\mathbb{N}}) &\equiv \emptyset \\ \mathsf{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}(n)) &\equiv \emptyset & \mathsf{Eq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n), \mathsf{succ}_{\mathbb{N}}(m)) &\equiv \mathsf{Eq}_{\mathbb{N}}(n, m). \end{aligned}$$

Construction. We define $\mathsf{Eq}_{\mathbb{N}}$ by double induction on $\mathbb{N}$. By the first application of induction it suffices to provide

$$\begin{aligned} E_0 &: \mathbb{N} \to \mathcal{U} \\ E_S &: \mathbb{N} \to (\mathbb{N} \to \mathcal{U}) \to (\mathbb{N} \to \mathcal{U}) \end{aligned}$$

We define E_0 by induction, taking $E_{00} :\equiv \mathbf{1}$ and $E_{0S}(n, X, m) :\equiv \emptyset$. The resulting family E_0 satisfies

$$\begin{aligned} E_0(0_{\mathbb{N}}) &\equiv \mathbf{1} \\ E_0(\mathsf{succ}_{\mathbb{N}}(n)) &\equiv \emptyset. \end{aligned}$$

We define E_S by induction, taking $E_{S0} :\equiv \emptyset$ and $E_{S0}(n, X, m) :\equiv X(m)$. The resulting family E_S satisfies

$$\begin{aligned} E_S(n, X, 0_{\mathbb{N}}) &\equiv \emptyset \\ E_S(n, X, \mathsf{succ}_{\mathbb{N}}(m)) &\equiv X(m) \end{aligned}$$

Therefore we have by the computation rule for the first induction that the judgmental equality

$$\begin{aligned} \mathsf{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, m) &\equiv E_0(m) \\ \mathsf{Eq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n), m) &\equiv E_S(n, \mathsf{Eq}_{\mathbb{N}}(n), m) \end{aligned}$$

holds, from which the judgmental equalities in the statement of the definition follow. □

Lemma 6.4.3. *Suppose $R : \mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ is a reflexive relation on $\mathbb{N}$, i.e., R comes equipped with*

$$\rho : \prod_{(n:\mathbb{N})} R(n, n).$$

Then there is a family of maps

$$\prod_{(m,n:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(m, n) \to R(m, n).$$

Proof. We will prove by induction on $m, n : \mathbb{N}$ that there is a term of type

$$f_{m,n} : \prod_{(e:\mathsf{Eq}_{\mathbb{N}}(m,n))} \prod_{(R:\mathbb{N}\to(\mathbb{N}\to\mathcal{U}))} \Big(\prod_{(x:\mathbb{N})} R(x, x)\Big) \to R(m, n)$$

The dependent function $f_{m,n}$ is defined by

$$\begin{aligned} f_{0_{\mathbb{N}}, 0_{\mathbb{N}}} &:\equiv \lambda \star.\, \lambda r.\, \lambda \rho.\, \rho(0_{\mathbb{N}}) \\ f_{0_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}(n)} &:\equiv \mathsf{ind}_{\emptyset} \\ f_{\mathsf{succ}_{\mathbb{N}}(m), 0_{\mathbb{N}}} &:\equiv \mathsf{ind}_{\emptyset} \end{aligned}$$

$$f_{\mathsf{succ}_{\mathbb{N}}(m),\mathsf{succ}_{\mathbb{N}}(n)} :\equiv \lambda e.\, \lambda R.\, \lambda \rho.\, f_{m,n}(e, R', \rho'),$$

where R' and ρ' are given by

$$\begin{aligned} R'(m,n) &:\equiv R(\mathsf{succ}_{\mathbb{N}}(m), \mathsf{succ}_{\mathbb{N}}(n)) \\ \rho'(n) &:\equiv \rho(\mathsf{succ}_{\mathbb{N}}(n)). \end{aligned}$$ □

We can also define observational equality for many other kinds of types, such as **2** or $\mathbb{Z}$. In each of these cases, what sets the observational equality apart from other relations is that it is the *least* reflexive relation.

Exercises

6.1 Show that observational equality on $\mathbb{N}$ is an equivalence relation, i.e., construct terms of the following types:

$$\prod_{(n:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(n,n)$$
$$\prod_{(n,m:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(n,m) \to \mathsf{Eq}_{\mathbb{N}}(m,n)$$
$$\prod_{(n,m,l:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(n,m) \to (\mathsf{Eq}_{\mathbb{N}}(m,l) \to \mathsf{Eq}_{\mathbb{N}}(n,l)).$$

6.2 Show that every function $f : \mathbb{N} \to \mathbb{N}$ preserves observational equality in the sense that

$$\prod_{(n,m:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(n,m) \to \mathsf{Eq}_{\mathbb{N}}(f(n), f(m)).$$

6.3 (a) Define the **order relations** $\leq$ and $<$ on $\mathbb{N}$.
(b) Show that $\leq$ is reflexive and that $<$ is **irreflexive**, i.e., show that

$$(n \leq n) \qquad \text{and} \qquad (n \not< n).$$

(c) Show that $n \leq \mathsf{succ}_{\mathbb{N}}(n)$ and $n < \mathsf{succ}_{\mathbb{N}}(n)$.
(d) Show that both $\leq$ and $<$ are transitive.
(e) Show that $\leq$ is **antisymmetric**, i.e., show that $m = n$ whenever both $m \leq n$ and $n \leq m$ hold. Also show that if $m < n$, then $n \not< m$.
(f) Show that

$$(m \leq n) \leftrightarrow (m = n) + (m < n)$$

holds for all m, n.
(g) Show that

$$(m \leq n) + (n < m)$$

holds for all m, n.
(h) Show that $k \leq \min(m,n)$ holds if and only if both $k \leq m$ and $k \leq n$ hold, and show that $\max(m,n) \leq k$ holds if and only if both $m \leq k$ and $n \leq k$ hold.

6.4 (a) Define observational equality $\mathsf{Eq}_{\mathbf{2}}$ on the booleans.
(b) Show that $\mathsf{Eq}_{\mathbf{2}}$ is reflexive.
(c) Show that for any reflexive relation $R : \mathbf{2} \to (\mathbf{2} \to \mathcal{U})$ one has

$$\prod_{(x,y:\mathbf{2})} \mathsf{Eq}_{\mathbf{2}}(x,y) \to R(x,y).$$

6.5 (a) Define the order relations $\leq$ and $<$ on and $\mathbb{Z}$.
(b) Show that $\leq$ is reflexive, transitive, and antisymmetric.
(c) Show that $<$ is irreflexive and transitive.

Chapter II

The univalent foundations for mathematics

In this chapter we study the foundational concepts of univalent mathematics. The first concept we study is that of equivalences. Equivalent types are the same for all practical purposes. The concept of equivalence generalizes the concept of isomorphism of sets to type theory. As we delve deeper into synthetic homotopy theory, we will see how vast this generalization really is, even though the type theoretic definition is really simple.

The next topic is that of contractible types and maps. Contractible types that are singletons up to homotopy, i.e., types that come equipped with a point so that every (other) point can be identified with it. Contractible maps are maps $f : A \to B$ such that for every $b : B$ the type $\sum_{(a:A)} f(a) = b$ is contractible. We will show that a map is an equivalence if and only if it is a contractible map in this sense.

It is a very important observation that for any $a : A$, the type

$$\textstyle\sum_{(x:A)} a = x$$

is contractible. In other words, the total space of the path fibration is contractible. The fundamental theorem of identity types asserts that a type family B over A with $b : B(a)$ has a contractible total space

$$\textstyle\sum_{(x:A)} B(x)$$

if and only if $(a = x) \simeq B(x)$ for all $x : A$. The fundamental theorem of identity types helps us characterizing the identity types of virtually any type that we will encounter. Since types are only fully understood if we also have a clear understanding of their identity types, it is one of the core tasks of a homotopy type theorist to characterize identity types and the fundamental theorem is the main tool.

Not all types have very complicated identity types. For example, some types have the property that all their identity types are contractible. Such types have up to homotopy at most one term, so they are in a sense “proof irrelevant”. The only thing that matters about such types is whether or not we can construct a term. Since this is also the case for propositions in first order logic, we call such types propositions.

At the next level there are types of which the identity types are propositions. In other words, the identity types of such types have the property of proof irrelevance. We are familiar with this situation from set theory, where equality is a proposition, so we call such types sets. One of our first major theorems is that the type of natural numbers is a set in this sense.

It is now clear that there is a hierarchy arising: first we have the contractible types; then we have the propositions, of which the identity types are contractible; after the propositions we have the sets, of which the identity types are propositions. Defining sets to be of truncation level 0, we define a type to be of truncation level $n+1$ if its identity types are of truncation level n.

The hierarchy of truncation level is due to Voevodsky, who recognized the importance of specifying the level which you are working in. Most mathematics, for example, takes place at truncation level 0, the level of sets. Groups, rings, posets, and so on are all set-level objects. Categories, on the other hand, are objects of truncation level 1, the level of groupoids. The study of all these objects will be greatly facilitated by the univalence axiom, so we will postpone a detailed discussion of them until then.

We end the chapter with a section on elementary number theory, the way it is done in type theory. Our goal is to show how to prove in type theory that there are infinitely many prime numbers. Here it will be important to show that the ordering and divisibility are properties, i.e., that the types $m \leq n$ and $d \mid n$ are propositions. Even more so, we will show that these are *decidable* propositions. Type theory is by its very nature a constructive theory, but that doesn't stop us from showing that either P or $\not P$ holds for some specific propositions P, like $m \leq n$ or $d \mid n$. One of the goals in the chapter on elementary number theory is to show how this is done, and how it is used.

7 Equivalences

We introduce equivalences in this section as functions that have a left inverse and a separate right inverse. This choice of definition might seem a little strange: why would we not say that an equivalence map is a map $f : A \to B$ for which there is a map $g : B \to A$ that is at the same time a left and a right inverse? We will be able to show, after all, that if a map has separate left and right inverses, then it has an inverse. For a precise answer to this question we will have to wait until Chapter III, but we can say already that it turns out to be important that the condition of being an equivalence is a property, not structure. We have chose the definition of equivalences with a separate left and right inverses so that being an equivalence will indeed be a property.

7.1 Homotopies

In homotopy type theory, a homotopy is just a pointwise equality between two functions f and g. We view the type of homotopies as the observational equality for Π-types.

Definition 7.1.1. Let $f, g : \prod_{(x:A)} P(x)$ be two dependent functions. The type of **homotopies** from f to g is defined as the type of pointwise identifications, i.e., we define

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

Note that the type of homotopies $f \sim g$ is a special case of a dependent function type. Therefore the definition of homotopies is set up in such a way that we may also consider homotopies *between* homotopies, and even further homotopies between those higher homotopies. More concretely, if $H, K : f \sim g$ are two homotopies, then the type of homotopies $H \sim K$ between them is just the type

$$\prod_{(x:A)} H(x) = K(x).$$

In the following definition we define the groupoidal structure of homotopies. Note that we implement the groupoid laws as *homotopies* rather than as identifications.

Definition 7.1.2. For any type family B over A there are operations

$$\mathsf{refl\text{-}htpy} : \prod_{(f:\prod_{(x:A)} B(x))} f \sim f$$

$$\mathsf{inv\text{-}htpy} : \prod_{(f,g:\prod_{(x:A)} B(x))} (f \sim g) \to (g \sim f)$$
$$\mathsf{concat\text{-}htpy} : \prod_{(f,g,h:\prod_{(x:A)} B(x))} (f \sim g) \to ((g \sim h) \to (f \sim h)).$$

We will write H^{-1} for $\mathsf{inv\text{-}htpy}(H)$, and $H \cdot K$ for $\mathsf{concat\text{-}htpy}(H, K)$.

Furthermore, we define

$$\begin{aligned}
\mathsf{assoc\text{-}htpy}(H, K, L) &: (H \cdot K) \cdot L \sim H \cdot (K \cdot L) \\
\mathsf{left\text{-}unit\text{-}htpy}(H) &: \mathsf{refl\text{-}htpy}_f \cdot H \sim H \\
\mathsf{right\text{-}unit\text{-}htpy}(H) &: H \cdot \mathsf{refl\text{-}htpy}_g \sim H \\
\mathsf{left\text{-}inv\text{-}htpy}(H) &: H^{-1} \cdot H \sim \mathsf{refl\text{-}htpy}_g \\
\mathsf{right\text{-}inv\text{-}htpy}(H) &: H \cdot H^{-1} \sim \mathsf{refl\text{-}htpy}_f
\end{aligned}$$

for any $H : f \sim g$, $K : g \sim h$ and $L : h \sim i$, where $f, g, h, i : \prod_{(x:A)} B(x)$.

Construction. We define

$$\begin{aligned}
\mathsf{refl\text{-}htpy}(f) &:\equiv \lambda x.\, \mathsf{refl}_{f(x)} \\
\mathsf{inv\text{-}htpy}(H) &:\equiv \lambda x.\, H(x)^{-1} \\
\mathsf{concat\text{-}htpy}(H, K) &:\equiv \lambda x.\, H(x) \cdot K(x),
\end{aligned}$$

where $H : f \sim g$ and $K : g \sim h$ are homotopies. Furthermore, we define

$$\begin{aligned}
\mathsf{assoc\text{-}htpy}(H, K, L) &:\equiv \lambda x.\, \mathsf{assoc}(H(x), K(x), L(x)) \\
\mathsf{left\text{-}unit\text{-}htpy}(H) &:\equiv \lambda x.\, \mathsf{left\text{-}unit}(H(x)) \\
\mathsf{right\text{-}unit\text{-}htpy}(H) &:\equiv \lambda x.\, \mathsf{right\text{-}unit}(H(x)) \\
\mathsf{left\text{-}inv\text{-}htpy}(H) &:\equiv \lambda x.\, \mathsf{left\text{-}inv}(H(x)) \\
\mathsf{right\text{-}inv\text{-}htpy}(h) &:\equiv \lambda x.\, \mathsf{right\text{-}inv}(H(x)).
\end{aligned}$$

□

Apart from the groupoid operations and their laws, we will occasionally need *whiskering* operations.

Definition 7.1.3. We define the following **whiskering** operations on homotopies:

(i) Suppose $H : f \sim g$ for two functions $f, g : A \to B$, and let $h : B \to C$. We define

$$h \cdot H :\equiv \lambda x.\, \mathsf{ap}_h(H(x)) : h \circ f \sim h \circ g.$$

(ii) Suppose $f : A \to B$ and $H : g \sim h$ for two functions $g, h : B \to C$. We define

$$H \cdot f :\equiv \lambda x.\, H(f(x)) : h \circ f \sim g \circ f.$$

We also use homotopies to express the commutativity of diagrams. For example, we say that a triangle

$$\begin{tikzcd}
A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\
& X &
\end{tikzcd}$$

commutes if it comes equipped with a homotopy $H : f \sim g \circ h$, and we say that a square

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

if it comes equipped with a homotopy $h \circ f \sim g \circ f'$.

7.2 Bi-invertible maps

Definition 7.2.1. Let $f : A \to B$ be a function. We say that f has a **section** if there is a term of type

$$\mathsf{sec}(f) :\equiv \textstyle\sum_{(g:B\to A)} f \circ g \sim \mathsf{id}_B.$$

Dually, we say that f has a **retraction** if there is a term of type

$$\mathsf{retr}(f) :\equiv \textstyle\sum_{(h:B\to A)} h \circ f \sim \mathsf{id}_A.$$

If a map $f : A \to B$ has a retraction, we also say that A is a **retract** of B. We say that a function $f : A \to B$ is an **equivalence** if it has both a section and a retraction, i.e., if it comes equipped with a term of type

$$\mathsf{is\text{-}equiv}(f) :\equiv \mathsf{sec}(f) \times \mathsf{retr}(f).$$

We will write $A \simeq B$ for the type $\sum_{(f:A\to B)} \mathsf{is\text{-}equiv}(f)$.

Remark 7.2.2. An equivalence, as we defined it here, can be thought of as a *bi-invertible map*, since it comes equipped with a separate left and right inverse. Explicitly, if f is an equivalence, then there are

$$\begin{array}{ll} g : B \to A & \qquad h : B \to A \\ G : f \circ g \sim \mathsf{id}_B & \qquad H : h \circ f \sim \mathsf{id}_A. \end{array}$$

Clearly, if f has an inverse in the sense that it comes equipped with a function $g : B \to A$ such that $f \circ g \sim \mathsf{id}_B$ and $g \circ f \sim \mathsf{id}_A$, then f is an equivalence. We write

$$\mathsf{has\text{-}inverse}(f) :\equiv \textstyle\sum_{(g:B\to A)} (f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A).$$

Lemma 7.2.3. *Any equivalence $e : A \simeq B$ can be given the structure of an invertible map. We define e^{-1} to be the section $g : B \to A$ of e.*

Proof. First we construct for any equivalence f with right inverse g and left inverse h a homotopy $K : g \sim h$. For any $y : B$, we have

$$g(y) \xlongequal{H(g(y))^{-1}} hfg(y) \xlongequal{\mathsf{ap}_h(G(y))} h(y).$$

Therefore we define a homotopy $K : g \sim h$ by $K :\equiv (H \cdot g)^{-1} \cdot h \cdot G$. Using the homotopy K we are able to show that g is also a left inverse of f. For $x : A$ we have the identification

$$gf(x) \xlongequal{K(f(x))} hf(x) \xlongequal{H(x)} x. \qquad \square$$

Corollary 7.2.4. *The inverse of an equivalence is again an equivalence.*

Proof. Let $f : A \to B$ be an equivalence. By Lemma 7.2.3 it follows that the section of f is also a retraction. Therefore it follows that the section is itself an invertible map, with inverse f. Hence it is an equivalence. □

Remark 7.2.5. For any type A, the identity function id_A is an equivalence, since it is its own section and its own retraction

Example 7.2.6. For any type $C(x, y)$ indexed by $x : A$ and $y : B$, the swap function

$$\sigma : \left(\prod_{(x:A)}\prod_{(y:B)}C(x,y)\right) \to \left(\prod_{(y:B)}\prod_{(x:A)}C(x,y)\right)$$

that swaps the order of the arguments x and y is an equivalence by Exercise 2.5.

7.3 The identity type of a Σ-type

In this section we characterize the identity type of a Σ-type as a Σ-type of identity types. In this course we will be characterizing the identity types of many types, so we will follow the general outline of how such a characterization goes:

(i) First we define a binary relation $R : A \to A \to \mathcal{U}$ on the type A that we are interested in. This binary relation is intended to be equivalent to its identity type.

(ii) Then we will show that this binary relation is reflexive, by constructing a term of type

$$\prod_{(x:A)}R(x,x)$$

(iii) Using the reflexivity we will show that there is a canonical map

$$(x = y) \to R(x,y)$$

for every $x, y : A$. This map is just constructed by path induction, using the reflexivity of R.

(iv) Finally, it has to be shown that the map

$$(x = y) \to R(x,y)$$

is an equivalence for each $x, y : A$.

The last step is usually the most difficult, and we will refine our methods for this step in §9, where we establish the fundamental theorem of identity types.

In this section we consider a type family B over A. Given two pairs

$$(x, y), (x', y') : \textstyle\sum_{(x:A)}B(x),$$

if we have a path $\alpha : x = x'$ then we can compare $y : B(x)$ to $y' : B(x')$ by first transporting y along α, i.e., we consider the identity type

$$\mathsf{tr}_B(\alpha, y) = y'.$$

Thus it makes sense to think of (x, y) to be identical to (x', y') if there is an identification $\alpha : x = x'$ and an identification $\beta : \mathsf{tr}_B(\alpha, y) = y'$. In the following definition we turn this idea into a binary relation on the Σ-type.

Definition 7.3.1. We will define a relation

$$\mathsf{Eq}_\Sigma : \Big(\textstyle\sum_{(x:A)} B(x)\Big) \to \Big(\textstyle\sum_{(x:A)} B(x)\Big) \to \mathcal{U}$$

by defining

$$\mathsf{Eq}_\Sigma(s,t) :\equiv \textstyle\sum_{(\alpha:\mathsf{pr}_1(s)=\mathsf{pr}_1(t))} \mathsf{tr}_B(\alpha, \mathsf{pr}_2(s)) = \mathsf{pr}_2(t).$$

Lemma 7.3.2. *The relation* Eq_Σ *is reflexive, i.e., there is a term*

$$\mathsf{reflexive\text{-}Eq}_\Sigma : \textstyle\prod_{(s:\sum_{(x:A)} B(x))} \mathsf{Eq}_\Sigma(s,s).$$

Construction. This term is constructed by Σ-induction on $s : \sum_{(x:A)} B(x)$. Thus, it suffices to construct a term of type

$$\textstyle\prod_{(x:A)} \prod_{(y:B(x))} \sum_{(\alpha:x=x)} \mathsf{tr}_B(\alpha, y) = y.$$

Here we take $\lambda x.\, \lambda y.\, (\mathsf{refl}_x, \mathsf{refl}_y)$. ∎

Definition 7.3.3. Consider a type family B over A. Then for any $s, t : \sum_{(x:A)} B(x)$ we define a map

$$\mathsf{pair\text{-}eq} : (s = t) \to \mathsf{Eq}_\Sigma(s,t)$$

by path induction, taking $\mathsf{pair\text{-}eq}(\mathsf{refl}_s) :\equiv \mathsf{reflexive\text{-}Eq}_\Sigma(s)$.

Theorem 7.3.4. *Let B be a type family over A. Then the map*

$$\mathsf{pair\text{-}eq} : (s = t) \to \mathsf{Eq}_\Sigma(s,t)$$

is an equivalence for every $s, t : \sum_{(x:A)} B(x)$.

Proof. The maps in the converse direction

$$\mathsf{eq\text{-}pair} : \mathsf{Eq}_\Sigma(s,t) \to (s = t)$$

are defined by repeated Σ-induction. By Σ-induction on s and t we see that it suffices to define a map

$$\mathsf{eq\text{-}pair} : \Big(\textstyle\sum_{(p:x=x')} \mathsf{tr}_B(p, y) = y'\Big) \to ((x,y) = (x',y')).$$

A map of this type is again defined by Σ-induction. Thus it suffices to define a dependent function of type

$$\textstyle\prod_{(p:x=x')} (\mathsf{tr}_B(p,y) = y') \to ((x,y) = (x',y')).$$

Such a dependent function is defined by double path induction by sending $(\mathsf{refl}_x, \mathsf{refl}_y)$ to $\mathsf{refl}_{(x,y)}$. This completes the definition of the function $\mathsf{eq\text{-}pair}$.

Next, we must show that $\mathsf{eq\text{-}pair}$ is a section of $\mathsf{pair\text{-}eq}$. In other words, we must construct an identification

$$\mathsf{pair\text{-}eq}(\mathsf{eq\text{-}pair}(\alpha, \beta)) = (\alpha, \beta)$$

for each $(\alpha, \beta) : \sum_{(\alpha:x=x')} \mathsf{tr}_B(\alpha, y) = y'$. We proceed by path induction on α, followed by path induction on β. Then our goal becomes to construct a term of type

$$\mathsf{pair\text{-}eq}(\mathsf{eq\text{-}pair}(\mathsf{refl}_x, \mathsf{refl}_y)) = (\mathsf{refl}_x, \mathsf{refl}_y)$$

By the definition of $\mathsf{eq\text{-}pair}$ we have $\mathsf{eq\text{-}pair}(\mathsf{refl}_x, \mathsf{refl}_y) \equiv \mathsf{refl}_{(x,y)}$, and by the definition of $\mathsf{pair\text{-}eq}$ we have $\mathsf{pair\text{-}eq}(\mathsf{refl}_{(x,y)}) \equiv (\mathsf{refl}_x, \mathsf{refl}_y)$. Thus we may take $\mathsf{refl}_{(\mathsf{refl}_x, \mathsf{refl}_y)}$ to complete the construction of the homotopy $\mathsf{pair\text{-}eq} \circ \mathsf{eq\text{-}pair} \sim \mathsf{id}$.

To complete the proof, we must show that $\mathsf{eq\text{-}pair}$ is a retraction of $\mathsf{pair\text{-}eq}$. In other words, we must construct an identification

$$\mathsf{eq\text{-}pair}(\mathsf{pair\text{-}eq}(p)) = p$$

for each $p : s = t$. We proceed by path induction on $p : s = t$, so it suffices to construct an identification

$$\mathsf{eq\text{-}pair}(\mathsf{refl}_{\mathsf{pr}_1(s)}, \mathsf{refl}_{\mathsf{pr}_2(s)}) = \mathsf{refl}_s.$$

Now we proceed by Σ-induction on $s : \sum_{(x:A)} B(x)$, so it suffices to construct an identification

$$\mathsf{eq\text{-}pair}(\mathsf{refl}_x, \mathsf{refl}_y) = \mathsf{refl}_{(x,y)}.$$

Since $\mathsf{eq\text{-}pair}(\mathsf{refl}_x, \mathsf{refl}_y)$ computes to $\mathsf{refl}_{(x,y)}$, we may simply take $\mathsf{refl}_{\mathsf{refl}_{(x,y)}}$. □

Exercises

7.1 Show that the functions

$$\begin{aligned} \mathsf{inv} &: (x = y) \to (y = x) \\ \mathsf{concat}(p) &: (y = z) \to (x = z) \\ \mathsf{concat}'(q) &: (x = y) \to (x = z) \\ \mathsf{tr}_B(p) &: B(x) \to B(y) \end{aligned}$$

are equivalences, where $\mathsf{concat}'(q, p) :\equiv p \cdot q$. Give their inverses explicitly.

7.2 Show that the maps

$$\begin{aligned} \mathsf{inl} &: X \to X + \emptyset & \qquad \mathsf{pr}_1 &: \emptyset \times X \to \emptyset \\ \mathsf{inr} &: X \to \emptyset + X & \qquad \mathsf{pr}_2 &: X \times \emptyset \to \emptyset \end{aligned}$$

are equivalences.

7.3 (a) Consider two functions $f, g : A \to B$ and a homotopy $H : f \sim g$. Then

$$\mathsf{is\text{-}equiv}(f) \leftrightarrow \mathsf{is\text{-}equiv}(g).$$

(b) Show that for any two homotopic equivalences $e, e' : A \simeq B$, their inverses are also homotopic.

7.4 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X. & \end{tikzcd}$$

with $H : f \sim g \circ h$.

(a) Suppose that the map h has a section $s : B \to A$. Show that the triangle

$$\begin{tikzcd} B \arrow[rr, "s"] \arrow[dr, "g"'] & & A \arrow[dl, "f"] \\ & X. & \end{tikzcd}$$

commutes, and that f has a section if and only if g has a section.

(b) Suppose that the map g has a retraction $r : X \to B$. Show that the triangle

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "h"'] & X \arrow[d, "r"] \\ & B. \end{tikzcd}$$

commutes, and that f has a retraction if and only if h has a retraction.

(c) (The **3-for-2 property** for equivalences.) Show that if any two of the functions

$$f, \qquad g, \qquad h$$

are equivalences, then so is the third.

7.5 (a) Show that the negation function on the booleans $\mathsf{neg}_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ defined in Example 4.4.2 is an equivalence.

(b) Use the observational equality on the booleans, defined in Exercise 6.4, to show that $0_{\mathbf{2}} \neq 1_{\mathbf{2}}$.

(c) Show that for any $b : \mathbf{2}$, the constant function const_b is not an equivalence.

7.6 Show that the successor function on the integers is an equivalence.

7.7 Construct a equivalences

$$A + B \simeq B + A \qquad \text{and} \qquad A \times B \simeq B \times A.$$

7.8 Consider a section-retraction pair

$$A \xrightarrow{i} B \xrightarrow{r} A,$$

with $H : r \circ i \sim \mathsf{id}$. Show that $x = y$ is a retract of $i(x) = i(y)$.

7.9 In this exercise we will show that the laws for abelian groups hold for addition on the integers. Note: these are obvious facts, but the proof terms that show *how* the group laws hold are nevertheless fairly involved. This exercise is perfect for a formalization project.

(a) Show that addition satisfies the left and right unit laws, i.e., show that

$$0 + x = x$$
$$x + 0 = x.$$

(b) Show that the following successor and predecessor laws hold for addition on $\mathbb{Z}$.

$$\mathsf{pred}_{\mathbb{Z}}(x) + y = \mathsf{pred}_{\mathbb{Z}}(x + y) \qquad \mathsf{succ}_{\mathbb{Z}}(x) + y = \mathsf{succ}_{\mathbb{Z}}(x + y)$$
$$x + \mathsf{pred}_{\mathbb{Z}}(y) = \mathsf{pred}_{\mathbb{Z}}(x + y) \qquad x + \mathsf{succ}_{\mathbb{Z}}(y) = \mathsf{succ}_{\mathbb{Z}}(x + y).$$

Hint: to avoid an excessive number of cases, use induction on x but not on y. You may need to use the homotopies $\mathsf{succ}_{\mathbb{Z}} \circ \mathsf{pred}_{\mathbb{Z}} \sim \mathsf{id}$ and $\mathsf{pred}_{\mathbb{Z}} \circ \mathsf{succ}_{\mathbb{Z}} \sim \mathsf{id}$ constructed in exercise Exercise 7.6.

(c) Use part (b) to show that addition on the integers is associative and commutative, show that

$$(x + y) + z = x + (y + z)$$
$$x + y = y + x.$$

Hint: Especially in the construction of the associator there is a risk of running into an unwieldy amount of cases if you use $\mathbb{Z}$-induction on all arguments. Avoid induction on y and z.

(d) Show that addition satisfies the left and right inverse laws:

$$(-x) + x = 0$$
$$x + (-x) = 0.$$

Conclude that the functions $y \mapsto x + y$ and $x \mapsto x + y$ are equivalences for any $x : \mathbb{Z}$ and $y : \mathbb{Z}$, respectively.

7.10 In this exercise we will show that $\mathbb{Z}$ satisfies the axioms of a **ring**.

(a) Show that multiplication on $\mathbb{Z}$ satisfies the following laws for 0 and 1:

$$0 \cdot x = 0 \qquad\qquad 1 \cdot x = x$$
$$x \cdot 0 = 0 \qquad\qquad x \cdot 1 = x.$$

(b) Show that multiplication on $\mathbb{Z}$ satisfies the predecessor and successor laws:

$$\mathsf{pred}_{\mathbb{Z}}(x) \cdot y = x \cdot y - y \qquad\qquad \mathsf{succ}_{\mathbb{Z}}(x) \cdot y = x \cdot y + y$$
$$x \cdot \mathsf{pred}_{\mathbb{Z}}(y) = x \cdot y - x \qquad\qquad y \cdot \mathsf{succ}_{\mathbb{Z}}(y) = x \cdot y + x.$$

(c) Show that multiplication on $\mathbb{Z}$ distributes over addition, both from the left and from the right:

$$x \cdot (y + z) = x \cdot y + x \cdot z$$
$$(x + y) \cdot z = x \cdot z + y \cdot z.$$

(d) Show that multiplication on Z is associative and commutative:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$
$$x \cdot y = y \cdot x.$$

7.11 In this exercise we will construct the **functorial action** of coproducts.

(a) Construct for any two maps $f : A \to A'$ and $g : B \to B'$, a map

$$f + g : A + B \to A' + B'.$$

(b) Show that if $H : f \sim f'$ and $K : g \sim g'$, then there is a homotopy

$$H + K : (f + g) \sim (f' + g').$$

(c) Show that $\mathsf{id}_A + \mathsf{id}_B \sim \mathsf{id}_{A+B}$.

(d) Show that for any

$$A \xrightarrow{f} A' \xrightarrow{f'} A''$$
$$B \xrightarrow{g} B' \xrightarrow{g'} B''$$

there is a homotopy

$$(f' \circ f) + (g' \circ g) \sim (f' + g') \circ (f \circ g).$$

(e) Show that if f and g are equivalences, then so is $f + g$. (The converse of this statement also holds, see Exercise 9.5.)

7.12 Construct equivalences

$$\begin{aligned}\mathsf{Fin}(m+n) &\simeq \mathsf{Fin}(m) + \mathsf{Fin}(n)\\ \mathsf{Fin}(mn) &\simeq \mathsf{Fin}(m) \times \mathsf{Fin}(n).\end{aligned}$$

8 Contractible types and contractible maps

A contractible type is a type which has, up to identification, only one term. In other words, a contractible type is a type that comes equipped with a point, and an identification of this point with any point.

We may think of contractible types as singletons up to homotopy, and indeed we show that the unit type is an example of a contractible type. Moreover, we show that contractible types satisfy an induction principle that is very similar to the induction principle of the unit type, provided that we formulate the computation rule using the identity type rather than postulating a judgmental computation rule.

Another case of an inductive type with a single constructor is the type of identifications $p : a = x$ with a fixed starting point $a : A$. To specify such an identification, we have to give its end point $x : A$ as well as the identification $p : a = x$, and the path induction principle asserts that in order to show something about all such identifications, it suffices to show that thing in the case where the end point is a, and the path is refl_a. This suggests that the total space

$$\textstyle\sum_{(x:A)} a = x$$

of all paths with starting point $a : A$ is contractible. This important fact will be shown in Theorem 8.1.7, and it is the basis for the fundamental theorem of identity types (§9).

In the remainder of this section we will show that for any equivalence $e : A \simeq B$ and any $b : B$, the type of all $a : A$ equipped with a path $p : e(a) = b$ is contractible. In other words, if a map $f : A \to B$ is an equivalence, then the 'preimage'

$$\textstyle\sum_{(a:A)} f(a) = b$$

is contractible for each $b : B$. The preimage of a map $f : A \to B$ at a point $b : B$ is called the fiber of f at b, and we say that a map is contractible if all its fibers are contractible. This condition is of course analogous to the set theoretic notion of bijective map, or 1-to-1-correspondence. We will see that a map is contractible if and only if it is an equivalence.

8.1 Contractible types

Definition 8.1.1. We say that a type A is **contractible** if it comes equipped with a term of type

$$\mathsf{is\text{-}contr}(A) :\equiv \textstyle\sum_{(c:A)} \prod_{(x:A)} c = x.$$

Given a term $(c, C) : \mathsf{is\text{-}contr}(A)$, we call $c : A$ the **center of contraction** of A, and we call $C : \prod_{(x:A)} c = x$ the **contraction** of A.

Remark 8.1.2. Suppose A is a contractible type with center of contraction c and contraction C. Then the type of C is (judgmentally) equal to the type

$$\mathsf{const}_c \sim \mathsf{id}_A.$$

In other words, the contraction C is a *homotopy* from the constant function to the identity function.

Example 8.1.3. The unit type is easily seen to be contractible. For the center of contraction we take $\star : \mathbf{1}$. Then we define a contraction $\prod_{(x:\mathbf{1})} \star = x$ by the induction principle of $\mathbf{1}$. Applying the induction principle, it suffices to construct a term of type $\star = \star$, for which we just take $\mathsf{refl}_\star$.

Definition 8.1.4. Suppose A comes equipped with a term $a : A$. Then we say that A satisfies **singleton induction** if for every type family B over A, the map

$$\mathsf{ev\text{-}pt} : \Big(\prod_{(x:A)} B(x)\Big) \to B(a)$$

defined by $\mathsf{ev\text{-}pt}(f) :\equiv f(a)$ has a section. In other words, if A satisfies singleton induction we have a function and a homotopy

$$\mathsf{ind\text{-}sing}_a : B(a) \to \prod_{(x:A)} B(x)$$
$$\mathsf{comp\text{-}sing}_a : \mathsf{ev\text{-}pt} \circ \mathsf{ind\text{-}sing}_a \sim \mathsf{id}$$

for any type family B over A.

Example 8.1.5. Note that the singleton induction principle is almost the same as the induction principle for the unit type, the difference being that the "computation rule" in the singleton induction for A is stated using an *identification* rather than as a judgmental equality. The unit type $\mathbf{1}$ comes equipped with a function

$$\mathsf{ind}_{\mathbf{1}} : B(\star) \to \prod_{(x:\mathbf{1})} B(x)$$

for every type family B over $\mathbf{1}$, satisfying the judgmental equality $\mathsf{ind}_{\mathbf{1}}(b, \star) \equiv b$ for every $b : B(\star)$ by the computation rule. Thus we easily obtain the homotopy

$$\lambda b.\, \mathsf{refl}_b : \mathsf{ev\text{-}pt} \circ \mathsf{ind}_{\mathbf{1}} \sim \mathsf{id},$$

and we conclude that the unit type satisfies singleton induction.

Theorem 8.1.6. *Let A be a type. The following are equivalent:*

(i) The type A is contractible.

(ii) The type A comes equipped with a term $a : A$, and satisfies singleton induction.

Proof. Suppose A is contractible with center of contraction a and contraction C. First we observe that, without loss of generality, we may assume that C comes equipped with an identification $p : C(a) = \mathsf{refl}_a$. To see this, note that we can always define a new contraction C' by

$$C'(x) :\equiv C(a)^{-1} \cdot C(x),$$

which satisfies the requirement by the left inverse law, constructed in Definition 5.2.5.

To show that A satisfies singleton induction let B be a type family over A, and suppose we have $b : B(a)$. Our goal is to define

$$\mathsf{ind\text{-}sing}_a(b) : \prod_{(x:A)} B(x).$$

Let $x : A$. Since we have an identification $C(x) : a = x$, and a term b in $B(a)$, we may transport b along the path $C(x)$ to obtain

$$\mathsf{ind\text{-}sing}_a(b, x) :\equiv \mathsf{tr}_B(C(x), b) : B(x).$$

Therefore, the function $\mathsf{ind\text{-}sing}_a(b)$ is defined to be the dependent function $\lambda x.\, \mathsf{tr}_B(C(x), b)$. Now we have to show that $\mathsf{ind\text{-}sing}_a(b, a) = b$. Then we have the identifications

$$\mathsf{tr}_B(C(a), b) \xlongequal{\mathsf{ap}_{\lambda\omega.\, \mathsf{tr}_B(\omega,b)}(p)} \mathsf{tr}_B(\mathsf{refl}_a, b) \xlongequal{\mathsf{refl}_b} b.$$

This shows that the computation rule is satisfied, which completes the proof that A satisfies singleton induction.

For the converse, suppose that $a : A$ and that A satisfies singleton induction. Our goal is to show that A is contractible. For the center of contraction we take the term $a : A$. By singleton induction applied to $B(x) :\equiv a = x$ we have the map

$$\mathsf{ind\text{-}sing}_a : a = a \to \prod_{(x:A)} a = x.$$

Therefore $\mathsf{ind\text{-}sing}_a(\mathsf{refl}_a)$ is a contraction. □

Theorem 8.1.7. *For any $a : A$, the type*

$$\sum_{(x:A)} a = x$$

is contractible.

Proof. We will prove the statement by showing that $\sum_{(y:A)} x = y$ satisfies singleton induction, and then use Theorem 8.1.6 to conclude that $\sum_{(x:A)} a = x$ is contractible. We will use the term $(a, \mathsf{refl}_a) : \sum_{(x:A)} a = x$ as the center of contraction.

Now let P be a type family over $\sum_{(x:A)} a = x$. Note that we have a commuting triangle

$$\begin{tikzcd} \prod_{(t:\sum_{(x:A)} a=x)} P(t) \arrow[rr, "\mathsf{ev\text{-}pair}"] \arrow[dr, "\mathsf{ev\text{-}pt}"'] & & \prod_{(x:A)} \prod_{(p:a=x)} P(x,p) \arrow[dl, "\mathsf{ev\text{-}refl}"] \\ & P(a, \mathsf{refl}_a) & \end{tikzcd}$$

where the maps $\mathsf{ev\text{-}pair}$ and $\mathsf{ev\text{-}refl}$ are defined as

$$f \mapsto \lambda x.\, \lambda p.\, f(x, p)$$
$$g \mapsto g(a, \mathsf{refl}_a),$$

respectively. By the induction principle for Σ-types it follows that $\mathsf{ev\text{-}pair}$ has a section, and by path induction it follows that $\mathsf{ev\text{-}refl}$ has a section. Therefore it follows from Exercise 7.4 that the composite $\mathsf{ev\text{-}pt}$ has a section. □

8.2 Contractible maps

Definition 8.2.1. Let $f : A \to B$ be a function, and let $b : B$. The **fiber** of f at b is defined to be the type

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

In other words, the fiber of f at b is the type of $a : A$ that get mapped by f to b. One may think of the fiber as a type theoretic version of the pre-image of a point.

It will be useful to have a characterization of the identity type of a fiber, so we will make such a characterization immediately.

Definition 8.2.2. Let $f : A \to B$ be a map, and let $(x, p), (x', p') : \mathsf{fib}_f(y)$ for some $y : B$. Then we define

$$\mathsf{Eq\text{-}fib}_f((x,p),(x',p')) :\equiv \textstyle\sum_{(\alpha:x=x')} p = \mathsf{ap}_f(\alpha) \cdot p'$$

The relation $\mathsf{Eq\text{-}fib}_f : \mathsf{fib}_f(y) \to \mathsf{fib}_f(y) \to \mathcal{U}$ is a reflexive relation, since we have

$$\lambda(x,p).\,(\mathsf{refl}_x, \mathsf{refl}_p) : \textstyle\prod_{((x,p):\mathsf{fib}_f(y))} \mathsf{Eq\text{-}fib}_f((x,p),(x,p)).$$

Lemma 8.2.3. *Consider a map $f : A \to B$ and let $y : B$. The canonical map*

$$((x,p) = (x',p')) \to \mathsf{Eq\text{-}fib}_f((x,p),(x',p'))$$

induced by the reflexivity of $\mathsf{Eq\text{-}fib}_f$ is an equivalence for any $(x,p),(x',p') : \mathsf{fib}_f(y)$.

Proof. The converse map

$$\mathsf{Eq\text{-}fib}_f((x,p),(x',p')) \to ((x,p) = (x',p'))$$

is easily defined by Σ-induction, and then path induction twice. The homotopies witnessing that this converse map is indeed a right inverse as well as a left inverse is similarly constructed by induction. □

Now we arrive at the notion of contractible map.

Definition 8.2.4. We say that a function $f : A \to B$ is **contractible** if there is a term of type

$$\mathsf{is\text{-}contr}(f) :\equiv \textstyle\prod_{(b:B)} \mathsf{is\text{-}contr}(\mathsf{fib}_f(b)).$$

Theorem 8.2.5. *Any contractible map is an equivalence.*

Proof. Let $f : A \to B$ be a contractible map. Using the center of contraction of each $\mathsf{fib}_f(y)$, we obtain a term of type

$$\lambda y.\,(g(y), G(y)) : \textstyle\prod_{(y:B)} \mathsf{fib}_f(y).$$

Thus, we get map $g : B \to A$, and a homotopy $G : \prod_{(y:B)} f(g(y)) = y$. In other words, we get a section of f.

It remains to construct a retraction of f. Taking g as our retraction, we have to show that $\prod_{(x:A)} g(f(x)) = x$. Note that we get an identification $p : f(g(f(x))) = f(x)$ since g is a section of f. It follows that $(g(f(x)), p) : \mathsf{fib}_f(f(x))$. Moreover, since $\mathsf{fib}_f(f(x))$ is contractible we get an identification $q : (g(f(x)), p) = (x, \mathsf{refl}_{f(x)})$. The base path $\mathsf{ap}_{\mathsf{pr}_1}(q)$ of this identification is an identification of type $g(f(x)) = x$, as desired. □

8.3 Equivalences are contractible maps

In Theorem 8.3.6 we will show the converse to Theorem 8.2.5, i.e., we will show that any equivalence is a contractible map. We will do this in two steps.

First we introduce a new notion of *coherently invertible map*, for which we can easily show that such maps have contractible fibers. Then we show that any equivalence is a coherently invertible map.

Recall that an invertible map is a map $f : A \to B$ equipped with $g : B \to A$ and homotopies

$$G : f \circ g \sim \mathsf{id} \qquad \text{and} \qquad H : g \circ f \sim \mathsf{id}.$$

Then we observe that both $G \cdot f$ and $f \cdot H$ are homotopies of the same type

$$f \circ g \circ f \sim f.$$

A coherently invertible map is an invertible map for which there is a further homotopy $G \cdot f \sim f \cdot H$.

Definition 8.3.1. Consider a map $f : A \to B$. We say that f is **coherently invertible** if it comes equipped with

$$\begin{aligned} g &: B \to A \\ G &: f \circ g \sim \mathsf{id} \\ H &: g \circ f \sim \mathsf{id} \\ K &: G \cdot f \sim f \cdot H. \end{aligned}$$

We will write $\mathsf{is\text{-}coh\text{-}invertible}(f)$ for the type of quadruples (g, G, H, K).

Although we will encounter the notion of coherently invertible map on some further occasions, the following lemma is our main motivation for considering it.

Lemma 8.3.2. *Any coherently invertible map has contractible fibers.*

Proof. Consider a map $f : A \to B$ equipped with

$$\begin{aligned} g &: B \to A \\ G &: f \circ g \sim \mathsf{id} \\ H &: g \circ f \sim \mathsf{id} \\ K &: G \cdot f \sim f \cdot H, \end{aligned}$$

and let $y : B$. Our goal is to show that $\mathsf{fib}_f(y)$ is contractible. For the center of contraction we take $(g(y), G(y))$. In order to construct a contraction, it suffices to construct a term of type

$$\prod_{(x:A)} \prod_{(p:f(x)=y)} \mathsf{Eq\text{-}fib}_f((g(y), G(y)), (x, p)).$$

By path induction on $p : f(x) = y$ it suffices to construct a term of type

$$\prod_{(x:A)} \mathsf{Eq\text{-}fib}_f((g(f(x)), G(f(x))), (x, \mathsf{refl}_{f(x)})).$$

By definition of $\mathsf{Eq\text{-}fib}_f$, we have to construct a term of type

$$\prod_{(x:A)}\sum_{(\alpha:g(f(x))=x)} G(f(x)) = \mathsf{ap}_f(\alpha) \cdot \mathsf{refl}_{f(x)}.$$

Such a term is constructed as $\lambda x.\,(H(x), K'(x))$, where the homotopy $H : g \circ f \sim \mathsf{id}$ is given by assumption, and the homotopy

$$K' : \prod_{(x:A)} G(f(x)) = \mathsf{ap}_f(H(x)) \cdot \mathsf{refl}_{f(x)}$$

is defined as

$$K' :\equiv K \cdot \mathsf{right\text{-}unit\text{-}htpy}(f \cdot H)^{-1}.$$

□

Our next goal is to show that for any map $f : A \to B$ equipped with

$$g : B \to A, \qquad G : f \circ g \sim \mathsf{id}, \qquad \text{and} \qquad H : g \circ f \sim \mathsf{id},$$

we can improve the homotopy G to a new homotopy $G' : f \circ g \sim \mathsf{id}$ for which there is a further homotopy

$$f \cdot H \sim G' \cdot f.$$

Note that this situation is analogous to the situation in the proof of Theorem 8.1.6, where we improved the contraction C so that it satisfied $C(c) = \mathsf{refl}$. The extra coherence $f \cdot H \sim G' \cdot f$ is then used in the proof that the fibers of an equivalence are contractible.

Definition 8.3.3. Let $f, g : A \to B$ be functions, and consider $H : f \sim g$ and $p : x = y$ in A. We define the identification

$$\mathsf{nat\text{-}htpy}(H, p) :\equiv \mathsf{ap}_f(p) \cdot H(y) = H(x) \cdot \mathsf{ap}_g(p)$$

witnessing that the square

$$\begin{tikzcd} f(x) \arrow[r, equal, "H(x)"] \arrow[d, equal, "\mathsf{ap}_f(p)"'] & g(x) \arrow[d, equal, "\mathsf{ap}_g(p)"] \\ f(y) \arrow[r, equal, "H(y)"'] & g(y) \end{tikzcd}$$

commutes. This square is also called the **naturality square** of the homotopy H at p.

Construction. By path induction on p it suffices to construct an identification

$$\mathsf{ap}_f(\mathsf{refl}_x) \cdot H(x) = H(x) \cdot \mathsf{ap}_g(\mathsf{refl}_x)$$

since $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$ and $\mathsf{ap}_g(\mathsf{refl}_x) \equiv \mathsf{refl}_{g(x)}$, and since $\mathsf{refl}_{f(x)} \cdot H(x) \equiv H(x)$, we see that the path $\mathsf{right\text{-}unit}(H(x))^{-1}$ is of the asserted type. □

Definition 8.3.4. Consider $f : A \to A$ and $H : f \sim \mathsf{id}_A$. We construct an identification $H(f(x)) = \mathsf{ap}_f(H(x))$, for any $x : A$.

Construction. By the naturality of homotopies with respect to identifications the square

$$\begin{tikzcd} ff(x) \arrow[r, equal, "H(f(x))"] \arrow[d, equal, "\mathsf{ap}_f(H(x))"'] & f(x) \arrow[d, equal, "H(x)"] \\ f(x) \arrow[r, equal, "H(x)"'] & x \end{tikzcd}$$

commutes. This gives the desired identification $H(f(x)) = \mathsf{ap}_f(H(x))$. □

Lemma 8.3.5. *Let $f : A \to B$ be a map, and consider $(g, G, H) : \mathsf{has\text{-}inverse}(f)$. Then there is a homotopy $G' : f \circ g \sim \mathsf{id}$ equipped with a further homotopy*

$$K : G' \cdot f \sim f \cdot H.$$

Thus we obtain a map $\mathsf{has\text{-}inverse}(f) \to \mathsf{is\text{-}coh\text{-}invertible}(f)$.

Proof. For each $y : B$, we construct the identification $G'(y)$ as the concatenation

$$fg(y) \xlongequal{G(fg(y))^{-1}} fgfg(y) \xlongequal{\mathsf{ap}_f(H(g(y)))} fg(y) \xlongequal{G(y)} y.$$

In order to construct a homotopy $G' \cdot f \sim f \cdot H$, it suffices to show that the square

$$\begin{tikzcd}
fgfgf(x) \arrow[r, equal, "G(fgf(x))"] \arrow[d, equal, "\mathsf{ap}_f(H(gf(x)))"'] & fgf(x) \arrow[d, equal, "\mathsf{ap}_f(H(x))"] \\
fgf(x) \arrow[r, equal, "G(f(x))"'] & f(x)
\end{tikzcd}$$

commutes for every $x : A$. Recall from Definition 8.3.4 that we have $H(gf(x)) = \mathsf{ap}_{gf}(H(x))$. Using this identification, we see that it suffices to show that the square

$$\begin{tikzcd}
fgfgf(x) \arrow[r, equal, "G(fgf(x))"] \arrow[d, equal, "\mathsf{ap}_{fgf}(H(x))"'] & fgf(x) \arrow[d, equal, "\mathsf{ap}_f(H(x))"] \\
fgf(x) \arrow[r, equal, "G(f(x))"'] & f(x)
\end{tikzcd}$$

commutes. Now we observe that this is just a naturality square the homotopy $Gf : fgf \sim f$, which commutes by Definition 8.3.3. □

Now we put the pieces together to conclude that any equivalence has contractible fibers.

Theorem 8.3.6. *Any equivalence is a contractible map.*

Proof. We have seen in Lemma 8.3.2 that any coherently invertible map is a contractible map. Moreover, any equivalence has the structure of an invertible map by Lemma 7.2.3, and any invertible map is coherently invertible by Lemma 8.3.5. □

Corollary 8.3.7. *Let A be a type, and let $a : A$. Then the type*

$$\textstyle\sum_{(x:A)} x = a$$

is contractible.

Proof. By Remark 7.2.5, the identity function is an equivalence. Therefore, the fibers of the identity function are contractible by Theorem 8.3.6. Note that $\sum_{(x:A)} x = a$ is exactly the fiber of id_A at $a : A$. □

Exercises

8.1 Show that if A is contractible, then for any $x, y : A$ the identity type $x = y$ is also contractible.

8.2 Suppose that A is a retract of B. Show that

$$\mathsf{is\text{-}contr}(B) \to \mathsf{is\text{-}contr}(A).$$

8.3 (a) Show that for any type A, the map $\mathsf{const}_\star : A \to \mathbf{1}$ is an equivalence if and only if A is contractible.
(b) Apply Exercise 7.4 to show that for any map $f : A \to B$, if any two of the three assertions
  (i) A is contractible
  (ii) B is contractible
  (iii) f is an equivalence
hold, then so does the third.

8.4 Show that for any two types A and B, the following are equivalent:
  (i) Both A and B are contractible.
  (ii) The type $A \times B$ is contractible.

8.5 Let A be a contractible type with center of contraction $a : A$. Furthermore, let B be a type family over A. Show that the map $y \mapsto (a, y) : B(a) \to \sum_{(x:A)} B(x)$ is an equivalence.

8.6 Let B be a family of types over A, and consider the projection map

$$\mathsf{pr}_1 : \left(\textstyle\sum_{(x:A)} B(x)\right) \to A.$$

Show that for any $a : A$, the map

$$\lambda((x, y), p).\, \mathsf{tr}_B(p, y) : \mathsf{fib}_{\mathsf{pr}_1}(a) \to B(a),$$

is an equivalence. Conclude that pr_1 is an equivalence if and only if each $B(a)$ is contractible.

8.7 Construct for any map $f : A \to B$ an equivalence $e : A \simeq \sum_{(y:B)} \mathsf{fib}_f(y)$ and a homotopy $H : f \sim \mathsf{pr}_1 \circ e$ witnessing that the triangle

$$\begin{tikzcd} A \arrow[rr, "e"] \arrow[dr, "f"'] & & \sum_{(y:B)} \mathsf{fib}_f(y) \arrow[dl, "\mathsf{pr}_1"] \\ & B & \end{tikzcd}$$

commutes. The projection $\mathsf{pr}_1 : (\sum_{(y:B)} \mathsf{fib}_f(y)) \to B$ is sometimes also called the **fibrant replacement** of f, because first projection maps are fibrations in the homotopy interpretation of type theory.

9 The fundamental theorem of identity types

For many types it is useful to have a characterization of their identity types. For example, we have used a characterization of the identity types of the fibers of a map in order to conclude that any equivalence is a contractible map. The fundamental theorem of identity types is our main tool to carry out such characterizations, and with the fundamental theorem it becomes a routine task to characterize an identity type whenever that is of interest.

In our first application of the fundamental theorem of identity types we show that any equivalence is an embedding. Embeddings are maps that induce equivalences on identity types, i.e., they are the homotopical analogue of injective maps. In our second application we characterize the identity types of coproducts.

Throughout the rest of this book we will encounter many more occasions to characterize identity types. For example, we will show in Theorem 10.2.6 that the identity type of the natural numbers is equivalent to its observational equality, and we will show in Theorem 19.5.2 that the loop space of the circle is equivalent to $\mathbb{Z}$.

In order to prove the fundamental theorem of identity types, we first prove the basic fact that a family of maps is a family of equivalences if and only if it induces an equivalence on total spaces.

9.1 Families of equivalences

Definition 9.1.1. Consider a family of maps

$$f : \prod_{(x:A)} B(x) \to C(x).$$

We define the map

$$\mathsf{tot}(f) : \sum_{(x:A)} B(x) \to \sum_{(x:A)} C(x)$$

by $\lambda(x,y).\,(x, f(x,y))$.

Lemma 9.1.2. *For any family of maps $f : \prod_{(x:A)} B(x) \to C(x)$ and any $t : \sum_{(x:A)} C(x)$, there is an equivalence*

$$\mathsf{fib}_{\mathsf{tot}(f)}(t) \simeq \mathsf{fib}_{f(\mathsf{pr}_1(t))}(\mathsf{pr}_2(t)).$$

Proof. For any $p : \mathsf{fib}_{\mathsf{tot}(f)}(t)$ we define $\varphi(t,p) : \mathsf{fib}_{\mathsf{pr}_1(t)}(\mathsf{pr}_2(t))$ by Σ-induction on p. Therefore it suffices to define $\varphi(t,(s,\alpha)) : \mathsf{fib}_{\mathsf{pr}_1(t)}(\mathsf{pr}_2(t))$ for any $s : \sum_{(x:A)} B(x)$ and $\alpha : \mathsf{tot}(f)(s) = t$. Now we proceed by path induction on α, so it suffices to define $\varphi(\mathsf{tot}(f)(s),(s,\mathsf{refl})) : \mathsf{fib}_{f(\mathsf{pr}_1(\mathsf{tot}(f)(s)))}(\mathsf{pr}_2(\mathsf{tot}(f)(s)))$. Finally, we use Σ-induction on s once more, so it suffices to define

$$\varphi((x,f(x,y)),((x,y),\mathsf{refl})) : \mathsf{fib}_{f(x)}(f(x,y)).$$

Now we take as our definition

$$\varphi((x,f(x,y)),((x,y),\mathsf{refl})) :\equiv (y,\mathsf{refl}).$$

For the proof that this map is an equivalence we construct a map

$$\psi(t) : \mathsf{fib}_{f(\mathsf{pr}_1(t))}(\mathsf{pr}_2(t)) \to \mathsf{fib}_{\mathsf{tot}(f)}(t)$$

equipped with homotopies $G(t) : \varphi(t) \circ \psi(t) \sim \mathsf{id}$ and $H(t) : \psi(t) \circ \varphi(t) \sim \mathsf{id}$. In each of these definitions we use Σ-induction and path induction all the way through, until an obvious choice of definition becomes apparent. We define $\psi(t)$, $G(t)$, and $H(t)$ as follows:

$$\begin{aligned} \psi((x,f(x,y)),(y,\mathsf{refl})) &:\equiv ((x,y),\mathsf{refl}) \\ G((x,f(x,y)),(y,\mathsf{refl})) &:\equiv \mathsf{refl} \\ H((x,f(x,y)),((x,y),\mathsf{refl})) &:\equiv \mathsf{refl}. \end{aligned}$$

□

Theorem 9.1.3. *Let $f : \prod_{(x:A)} B(x) \to C(x)$ be a family of maps. The following are equivalent:*

(i) For each $x : A$, the map $f(x)$ is an equivalence. In this case we say that f is a ***family of equivalences****.*

(ii) The map $\mathsf{tot}(f) : \sum_{(x:A)} B(x) \to \sum_{(x:A)} C(x)$ is an equivalence.

Proof. By Theorems 8.2.5 and 8.3.6 it suffices to show that $f(x)$ is a contractible map for each $x : A$, if and only if $\mathsf{tot}(f)$ is a contractible map. Thus, we will show that $\mathsf{fib}_{f(x)}(c)$ is contractible if and only if $\mathsf{fib}_{\mathsf{tot}(f)}(x, c)$ is contractible, for each $x : A$ and $c : C(x)$. However, by Lemma 9.1.2 these types are equivalent, so the result follows by Exercise 8.3. □

Now consider the situation where we have a map $f : A \to B$, and a family C over B. Then we have the map

$$\lambda(x, z).\,(f(x), z) : \textstyle\sum_{(x:A)} C(f(x)) \to \sum_{(y:B)} C(y).$$

We claim that this map is an equivalence when f is an equivalence. The technique to prove this claim is the same as the technique we used in Theorem 9.1.3: first we note that the fibers are equivalent to the fibers of f, and then we use the fact that a map is an equivalence if and only if its fibers are contractible to finish the proof.

The converse of the following lemma does not hold. Why not?

Lemma 9.1.4. *Consider an equivalence $e : A \simeq B$, and let C be a type family over B. Then the map*

$$\sigma_f(C) :\equiv \lambda(x, z).\,(f(x), z) : \textstyle\sum_{(x:A)} C(f(x)) \to \sum_{(y:B)} C(y)$$

is an equivalence.

Proof. We claim that for each $t : \sum_{(y:B)} C(y)$ there is an equivalence

$$\mathsf{fib}_{\sigma_f(C)}(t) \simeq \mathsf{fib}_f(\mathsf{pr}_1(t)).$$

We obtain such an equivalence by constructing the following functions and homotopies:

$$\begin{aligned}
&\varphi(t) : \mathsf{fib}_{\sigma_f(C)}(t) \to \mathsf{fib}_f(\mathsf{pr}_1(t)) && \varphi((f(x), z), ((x, z), \mathsf{refl})) :\equiv (x, \mathsf{refl}) \\
&\psi(t) : \mathsf{fib}_f(\mathsf{pr}_1(t)) \to \mathsf{fib}_{\sigma_f(C)}(t) && \psi((f(x), z), (x, \mathsf{refl})) :\equiv ((x, z), \mathsf{refl}) \\
&G(t) : \varphi(t) \circ \psi(t) \sim \mathsf{id} && G((f(x), z), (x, \mathsf{refl})) :\equiv \mathsf{refl} \\
&H(t) : \psi(t) \circ \varphi(t) \sim \mathsf{id} && H((f(x), z), ((x, z), \mathsf{refl})) :\equiv \mathsf{refl}.
\end{aligned}$$

Now the claim follows, since we see that φ is a contractible map if and only if f is a contractible map. □

We now combine Theorem 9.1.3 and Lemma 9.1.4.

Definition 9.1.5. Consider a map $f : A \to B$ and a family of maps

$$g : \textstyle\prod_{(x:A)} C(x) \to D(f(x)),$$

where C is a type family over A, and D is a type family over B. In this situation we also say that g is a **family of maps over** f. Then we define

$$\mathsf{tot}_f(g) : \textstyle\sum_{(x:A)} C(x) \to \sum_{(y:B)} D(y)$$

by $\mathsf{tot}_f(g)(x, z) :\equiv (f(x), g(x, z))$.

Theorem 9.1.6. *Suppose that g is a family of maps over f, and suppose that f is an equivalence. Then the following are equivalent:*

(i) *The family of maps g over f is a family of equivalences.*

(ii) *The map $\mathsf{tot}_f(g)$ is an equivalence.*

Proof. Note that we have a commuting triangle

$$\begin{tikzcd} \sum_{(x:A)} C(x) \arrow[rr, "\mathsf{tot}_f(g)"] \arrow[dr, "\mathsf{tot}(g)"'] & & \sum_{(y:B)} D(y) \\ & \sum_{(x:A)} D(f(x)) \arrow[ur, "\lambda(x,z).\,(f(x),z)"'] & \end{tikzcd}$$

By the assumption that f is an equivalence, it follows that the map $\sum_{(x:A)} D(f(x)) \to \sum_{(y:B)} D(y)$ is an equivalence. Therefore it follows that $\mathsf{tot}_f(g)$ is an equivalence if and only if $\mathsf{tot}(g)$ is an equivalence. Now the claim follows, since $\mathsf{tot}(g)$ is an equivalence if and only if g if a family of equivalences. □

9.2 The fundamental theorem

Many types come equipped with a reflexive relation that possesses a similar structure as the identity type. The observational equality on the natural numbers is such an example. We have see that it is a reflexive, symmetric, and transitive relation, and moreover it is contained in any other reflexive relation. Thus, it is natural to ask whether observational equality on the natural numbers is equivalent to the identity type.

The fundamental theorem of identity types (Theorem 9.2.2) is a general theorem that can be used to answer such questions. It describes a necessary and sufficient condition on a type family B over a type A equipped with a point $a : A$, for there to be a family of equivalences $\prod_{(x:A)}(a = x) \simeq B(x)$. In other words, it tells us when a family B is a characterization of the identity type of A.

Before we state the fundamental theorem of identity types we introduce the notion of *identity systems*. Those are families B over a A that satisfy an induction principle that is similar to the path induction principle, where the 'computation rule' is stated with an identification.

Definition 9.2.1. Let A be a type equipped with a term $a : A$. A **(unary) identity system** on A at a consists of a type family B over A equipped with $b : B(a)$, such that for any family of types $P(x, y)$ indexed by $x : A$ and $y : B(x)$, the function

$$h \mapsto h(a, b) : \Big(\prod_{(x:A)}\prod_{(y:B(x))} P(x, y)\Big) \to P(a, b)$$

has a section.

The most important implication in the fundamental theorem is that (ii) implies (i). Occasionally we will also use the third equivalent statement. We note that the fundamental theorem also appears as Theorem 5.8.4 in [3].

Theorem 9.2.2. *Let A be a type with $a : A$, and let B be be a type family over A with $b : B(a)$. Then the following are logically equivalent for any family of maps*

$$f : \prod_{(x:A)}(a = x) \to B(x).$$

(i) The family of maps f is a family of equivalences.

(ii) The total space

$$\sum_{(x:A)} B(x)$$

is contractible.

(iii) The family B is an identity system.

In particular the canonical family of maps

$$\mathsf{path\text{-}ind}_a(b) : \prod_{(x:A)} (a = x) \to B(x)$$

is a family of equivalences if and only if $\sum_{(x:A)} B(x)$ is contractible.

Proof. First we show that (i) and (ii) are equivalent. By Theorem 9.1.3 it follows that the family of maps f is a family of equivalences if and only if it induces an equivalence

$$\Big(\sum_{(x:A)} a = x\Big) \simeq \Big(\sum_{(x:A)} B(x)\Big)$$

on total spaces. We have that $\sum_{(x:A)} a = x$ is contractible. Now it follows by Exercise 8.3, applied in the case

$$\begin{tikzcd} \sum_{(x:A)} a = x \arrow[rr, "\mathsf{tot}(f)"] \arrow[dr, "\simeq"'] & & \sum_{(x:A)} B(x) \arrow[dl] \\ & \mathbf{1} & \end{tikzcd}$$

that $\mathsf{tot}(f)$ is an equivalence if and only if $\sum_{(x:A)} B(x)$ is contractible.

Now we show that (ii) and (iii) are equivalent. Note that we have the following commuting triangle

$$\begin{tikzcd} \prod_{(t:\sum_{(x:A)} B(x))} P(t) \arrow[rr, "\mathsf{ev\text{-}pair}"] \arrow[dr, "\mathsf{ev\text{-}pt}(a,b)"'] & & \prod_{(x:A)} \prod_{(y:B(x))} P(x,y) \arrow[dl, "\lambda h.\, h(a,b)"] \\ & P(a,b) & \end{tikzcd}$$

In this diagram the top map has a section. Therefore it follows by Exercise 7.4 that the left map has a section if and only if the right map has a section. Notice that the left map has a section for all P if and only if $\sum_{(x:A)} B(x)$ satisfies singleton induction, which is by Theorem 8.1.6 equivalent to $\sum_{(x:A)} B(x)$ being contractible. □

9.3 Embeddings

As an application of the fundamental theorem we show that equivalences are embeddings. The notion of embedding is the homotopical analogue of the set theoretic notion of injective map.

Definition 9.3.1. An **embedding** is a map $f : A \to B$ satisfying the property that

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

is an equivalence for every $x, y : A$. We write $\mathsf{is\text{-}emb}(f)$ for the type of witnesses that f is an embedding.

Another way of phrasing the following statement is that equivalent types have equivalent identity types.

Theorem 9.3.2. *Any equivalence is an embedding.*

Proof. Let $e : A \simeq B$ be an equivalence, and let $x : A$. Our goal is to show that

$$\mathsf{ap}_e : (x = y) \to (e(x) = e(y))$$

is an equivalence for every $y : A$. By Theorem 9.2.2 it suffices to show that

$$\textstyle\sum_{(y:A)} e(x) = e(y)$$

is contractible for every $y : A$. Now observe that there is an equivalence

$$\begin{aligned}\textstyle\sum_{(y:A)} e(x) = e(y) &\simeq \textstyle\sum_{(y:A)} e(y) = e(x) \\ &\equiv \mathsf{fib}_e(e(x))\end{aligned}$$

by Theorem 9.1.3, since for each $y : A$ the map

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

is an equivalence by Exercise 7.1. The fiber $\mathsf{fib}_e(e(x))$ is contractible by Theorem 8.3.6, so it follows by Exercise 8.3 that the type $\sum_{(y:A)} e(x) = e(y)$ is indeed contractible. □

9.4 Disjointness of coproducts

To give a second application of the fundamental theorem of identity types, we characterize the identity types of coproducts. Our goal in this section is to prove the following theorem.

Theorem 9.4.1. *Let A and B be types. Then there are equivalences*

$$\begin{aligned}(\mathsf{inl}(x) = \mathsf{inl}(x')) &\simeq (x = x') \\ (\mathsf{inl}(x) = \mathsf{inr}(y')) &\simeq \emptyset \\ (\mathsf{inr}(y) = \mathsf{inl}(x')) &\simeq \emptyset \\ (\mathsf{inr}(y) = \mathsf{inr}(y')) &\simeq (y = y')\end{aligned}$$

for any $x, x' : A$ and $y, y' : B$.

In order to prove Theorem 9.4.1, we first define a binary relation $\mathsf{Eq\text{-}coprod}_{A,B}$ on the coproduct $A + B$.

Definition 9.4.2. Let A and B be types. We define

$$\mathsf{Eq\text{-}coprod}_{A,B} : (A + B) \to (A + B) \to \mathcal{U}$$

by double induction on the coproduct, postulating

$$\begin{aligned}\mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), \mathsf{inl}(x')) &:\equiv (x = x') \\ \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), \mathsf{inr}(y')) &:\equiv \emptyset\end{aligned}$$

$$\mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inr}(y), \mathsf{inl}(x')) :\equiv \emptyset$$
$$\mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inr}(y), \mathsf{inr}(y')) :\equiv (y = y')$$

The relation $\mathsf{Eq\text{-}coprod}_{A,B}$ is also called the **observational equality of coproducts**.

Lemma 9.4.3. *The observational equality relation* $\mathsf{Eq\text{-}coprod}_{A,B}$ *on* $A + B$ *is reflexive, and therefore there is a map*

$$\mathsf{Eq\text{-}coprod\text{-}eq} : \prod_{(s,t:A+B)} (s = t) \to \mathsf{Eq\text{-}coprod}_{A,B}(s, t)$$

Construction. The reflexivity term ρ is constructed by induction on $t : A + B$, using

$$\rho(\mathsf{inl}(x)) :\equiv \mathsf{refl}_{\mathsf{inl}(x)} : \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x))$$
$$\rho(\mathsf{inr}(y)) :\equiv \mathsf{refl}_{\mathsf{inr}(y)} : \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inr}(y)).$$ □

To show that Eq-coprod-eq is a family of equivalences, we will use the fundamental theorem, Theorem 9.2.2. Moreover, we will use the functoriality of coproducts (established in Exercise 7.11), and the fact that any total space over a coproduct is again a coproduct:

$$\sum_{(t:A+B)} P(t) \simeq \Big(\sum_{(x:A)} P(\mathsf{inl}(x))\Big) + \Big(\sum_{(y:B)} P(\mathsf{inr}(y))\Big)$$

All of these equivalences are straightforward to construct, so we leave them as an exercise to the reader.

Lemma 9.4.4. *For any* $s : A + B$ *the total space*

$$\sum_{(t:A+B)} \mathsf{Eq\text{-}coprod}_{A,B}(s, t)$$

is contractible.

Proof. We will do the proof by induction on s. The two cases are similar, so we only show that the total space

$$\sum_{(t:A+B)} \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), t)$$

is contractible. Note that we have equivalences

$$\begin{aligned}
&\sum_{(t:A+B)} \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), t) \\
&\simeq \Big(\sum_{(x':A)} \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), \mathsf{inl}(x'))\Big) + \Big(\sum_{(y':B)} \mathsf{Eq\text{-}coprod}_{A,B}(\mathsf{inl}(x), \mathsf{inr}(y'))\Big) \\
&\simeq \Big(\sum_{(x':A)} x = x'\Big) + \Big(\sum_{(y':B)} \emptyset\Big) \\
&\simeq \Big(\sum_{(x':A)} x = x'\Big) + \emptyset \\
&\simeq \sum_{(x':A)} x = x'.
\end{aligned}$$

In the last two equivalences we used Exercise 7.2. This shows that the total space is contractible, since the latter type is contractible by Theorem 8.1.7. □

Proof of Theorem 9.4.1. The proof is now concluded with an application of Theorem 9.2.2, using Lemma 9.4.4. □

Exercises

9.1 (a) Show that the map $\emptyset \to A$ is an embedding for every type A.
(b) Show that $\mathsf{inl} : A \to A + B$ and $\mathsf{inr} : B \to A + B$ are embeddings for any two types A and B.

9.2 Consider an equivalence $e : A \simeq B$. Construct an equivalence

$$(e(x) = y) \simeq (x = e^{-1}(y))$$

for every $x : A$ and $y : B$.

9.3 Show that

$$(f \sim g) \to (\mathsf{is\text{-}emb}(f) \leftrightarrow \mathsf{is\text{-}emb}(g))$$

for any $f, g : A \to B$.

9.4 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$.
(a) Suppose that g is an embedding. Show that f is an embedding if and only if h is an embedding.
(b) Suppose that h is an equivalence. Show that f is an embedding if and only if g is an embedding.

9.5 Consider two maps $f : A \to A'$ and $g : B \to B'$.
(a) Show that if the map

$$f + g : (A + B) \to (A' + B')$$

is an equivalence, then so are both f and g (this is the converse of Exercise 7.11.e).
(b) Show that $f + g$ is an embedding if and only if both f and g are embeddings.

9.6 (a) Let $f, g : \prod_{(x:A)} B(x) \to C(x)$ be two families of maps. Show that

$$\Big(\prod_{(x:A)} f(x) \sim g(x)\Big) \to \Big(\mathsf{tot}(f) \sim \mathsf{tot}(g)\Big).$$

(b) Let $f : \prod_{(x:A)} B(x) \to C(x)$ and let $g : \prod_{(x:A)} C(x) \to D(x)$. Show that

$$\mathsf{tot}(\lambda x.\, g(x) \circ f(x)) \sim \mathsf{tot}(g) \circ \mathsf{tot}(f).$$

(c) For any family B over A, show that

$$\mathsf{tot}(\lambda x.\, \mathsf{id}_{B(x)}) \sim \mathsf{id}.$$

9.7 Let $a : A$, and let B be a type family over A.
(a) Use Exercises 8.2 and 9.6 to show that if each $B(x)$ is a retract of $a = x$, then $B(x)$ is equivalent to $a = x$ for every $x : A$.
(b) Conclude that for any family of maps

$$f : \prod_{(x:A)} (a = x) \to B(x),$$

if each $f(x)$ has a section, then f is a family of equivalences.

9.8 Use Exercise 9.7 to show that for any map $f : A \to B$, if

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

has a section for each $x, y : A$, then f is an embedding.

9.9 We say that a map $f : A \to B$ is **path-split** if f has a section, and for each $x, y : A$ the map

$$\mathsf{ap}_f(x, y) : (x = y) \to (f(x) = f(y))$$

also has a section. We write $\mathsf{path\text{-}split}(f)$ for the type

$$\mathsf{sec}(f) \times \prod_{(x,y:A)} \mathsf{sec}(\mathsf{ap}_f(x, y)).$$

Show that for any map $f : A \to B$ the following are equivalent:

(i) The map f is an equivalence.

(ii) The map f is path-split.

9.10 Consider a triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with a homotopy $H : f \sim g \circ h$ witnessing that the triangle commutes.

(a) Construct a family of maps

$$\mathsf{fib\text{-}triangle}(h, H) : \prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(x),$$

for which the square

$$\begin{tikzcd} \sum_{(x:X)} \mathsf{fib}_f(x) \arrow[r, "\mathsf{tot}(\mathsf{fib\text{-}triangle}(h,H))"] \arrow[d] & \sum_{(x:X)} \mathsf{fib}_g(x) \arrow[d] \\ A \arrow[r, "h"'] & B \end{tikzcd}$$

commutes, where the vertical maps are as constructed in Exercise 8.7.

(b) Show that h is an equivalence if and only if $\mathsf{fib\text{-}triangle}(h, H)$ is a family of equivalences.

10 Propositions, sets, and the higher truncation levels

10.1 Propositions and subtypes

Definition 10.1.1. A type A is said to be a **proposition** if there is a term of type

$$\mathsf{is\text{-}prop}(A) :\equiv \prod_{(x,y:A)} \mathsf{is\text{-}contr}(x = y).$$

Given a universe $\mathcal{U}$, we define $\mathsf{Prop}_{\mathcal{U}}$ to be the type of all small propositions, i.e.,

$$\mathsf{Prop}_{\mathcal{U}} :\equiv \sum_{(X:\mathcal{U})} \mathsf{is\text{-}prop}(X).$$

Example 10.1.2. Any contractible type is a proposition by Exercise 8.1. In particular, the unit type is a proposition.

However, propositions do not need to be inhabited: the empty type is also a proposition, since

$$\prod_{(x,y:\emptyset)} \mathsf{is\text{-}contr}(x = y)$$

follows from the induction principle of the empty type.

In the following lemma we prove that in order to show that a type A is a proposition, it suffices to show that any two terms of A are equal. In other words, propositions are types with **proof irrelevance**.

Theorem 10.1.3. *Let A be a type. Then the following are equivalent:*

(i) The type A is a proposition.

(ii) Any two terms of type A can be identified, i.e., there is a dependent function

$$\mathsf{is\text{-}prop}'(A) :\equiv \prod_{(x,y:A)} x = y.$$

(iii) The type A is contractible as soon as it is inhabited, i.e., there is a function

$$A \to \mathsf{is\text{-}contr}(A).$$

(iv) The map $\mathsf{const}_\star : A \to \mathbf{1}$ is an embedding.

Proof. To show that (i) implies (ii), let A be a proposition. Then its identity types are contractible, so the center of contraction of $x = y$ is identification $x = y$, for each $x, y : A$.

To show that (ii) implies (iii), suppose that A comes equipped with $p : \prod_{(x,y:A)} x = y$. Then for any $x : A$ the dependent function $p(x) : \prod_{(y:A)} x = y$ is a contraction of A. Thus we obtain the function

$$\lambda x.\,(x, p(x)) : A \to \mathsf{is\text{-}contr}(A).$$

To show that (iii) implies (iv), suppose that $A \to \mathsf{is\text{-}contr}(A)$. We first make the simple observation that

$$(X \to \mathsf{is\text{-}emb}(f)) \to \mathsf{is\text{-}emb}(f)$$

for any map $f : X \to Y$, so it suffices to show that $A \to \mathsf{is\text{-}emb}(\mathsf{const}_\star)$. However, assuming we have $x : A$, it follows by assumption that A is contractible. Therefore, it follows by Exercise 8.3 that the map $\mathsf{const}_\star : A \to \mathbf{1}$ is an equivalence. Since it is an equivalence, it is an embedding by Theorem 9.3.2.

To show that (iv) implies (i), note that if $A \to \mathbf{1}$ is an embedding, then the identity types of A are equivalent to contractible types and therefore they must be contractible. ☐

In the following lemma we show that propositions are closed under equivalences.

Lemma 10.1.4. *Let A and B be types, and let $e : A \simeq B$. Then we have*

$$\mathsf{is\text{-}prop}(A) \leftrightarrow \mathsf{is\text{-}prop}(B).$$

Proof. We will show that $\mathsf{is\text{-}prop}(B)$ implies $\mathsf{is\text{-}prop}(A)$. This suffices, because the converse follows from the fact that $e^{-1} : B \to A$ is also an equivalence.

Since e is assumed to be an equivalence, it follows by Theorem 9.3.2 that

$$\mathsf{ap}_e : (x = y) \to (e(x) = e(y))$$

is an equivalence for any $x, y : A$. If B is a proposition, then in particular the type $e(x) = e(y)$ is contractible for any $x, y : A$, so the claim follows from Theorem 8.3.6. ☐

In set theory, a set y is said to be a subset of a set x, if any element of y is an element of x, i.e., if the condition

$$\forall_z (z \in y) \to (z \in x)$$

holds. We have already noted that type theory is different from set theory in that terms in type theory come equipped with a *unique* type. Moreover, in set theory the proposition $x \in y$ is well-formed for any two sets x and y, whereas in type theory the judgment $a : A$ is only well-formed if it is derived using the postulated inference rules. Because of these differences we must find a different way to talk about subtypes.

Note that in set theory there is a correspondence between the subsets of a set x, and the *predicates* on x. A predicate on x is just a proposition $P(z)$ that varies over the elements $z \in x$. Indeed, if y is a subset of x, then the corresponding predicate is the proposition $z \in y$. Conversely, if P is a predicate on x, then we obtain the subset

$$\{z \in x \mid P(z)\}$$

of x. Now we have the right idea of subtypes in type theory: they are families of propositions.

Definition 10.1.5. A type family B over A is said to be a **subtype** of A if for each $x : A$ the type $B(x)$ is a proposition. When B is a subtype of A, we also say that $B(x)$ is a **property** of $x : A$.

We will show in Corollary 10.3.8 that a type family B over A is a subtype of A if and only if the projection map $\mathsf{pr}_1 : \left(\sum_{(x:A)} B(x)\right) \to A$ is an embedding.

10.2 Sets

Definition 10.2.1. A type A is said to be a **set** if it comes equipped with a term of type

$$\mathsf{is\text{-}set}(A) :\equiv \prod_{(x,y:A)} \mathsf{is\text{-}prop}(x = y).$$

Lemma 10.2.2. *A type A is a set if and only if it satisfies **axiom K**, i.e., if and only if it comes equipped with a term of type*

$$\mathsf{axiom\text{-}K}(A) :\equiv \prod_{(x:A)} \prod_{(p:x=x)} \mathsf{refl}_x = p.$$

Proof. If A is a set, then $x = x$ is a proposition, so any two of its elements are equal. This implies axiom K.

For the converse, if A satisfies axiom K, then for any $p, q : x = y$ we have $p \cdot q^{-1} = \mathsf{refl}_x$, and hence $p = q$. This shows that $x = y$ is a proposition, and hence that A is a set. $\square$

Theorem 10.2.3. *Let A be a type, and let $R : A \to A \to \mathcal{U}$ be a binary relation on A satisfying*

(i) Each $R(x, y)$ is a proposition,

(ii) R is reflexive, as witnessed by $\rho : \prod_{(x:A)} R(x, x)$,

(iii) There is a map

$$R(x, y) \to (x = y)$$

for each $x, y : A$.

Then any family of maps

$$\prod_{(x,y:A)} (x = y) \to R(x, y)$$

is a family of equivalences. Consequently, the type A is a set.

Proof. Let $f : \prod_{(x,y:A)} R(x,y) \to (x = y)$. Since R is assumed to be reflexive, we also have a family of maps

$$\mathsf{path\text{-}ind}_x(\rho(x)) : \prod_{(y:A)}(x = y) \to R(x,y).$$

Since each $R(x,y)$ is assumed to be a proposition, it therefore follows that each $R(x,y)$ is a retract of $x = y$. Therefore it follows that $\sum_{(y:A)} R(x,y)$ is a retract of $\sum_{(y:A)} x = y$, which is contractible. We conclude that $\sum_{(y:A)} R(x,y)$ is contractible, and therefore that any family of maps

$$\prod_{(y:A)}(x = y) \to R(x,y)$$

is a family of equivalences.

Now it also follows that A is a set, since its identity types are equivalent to propositions, and therefore they are propositions by Lemma 10.1.4. □

Definition 10.2.4. A map $f : A \to B$ is said to be **injective** if for any $x, y : A$ there is a map

$$(f(x) = f(y)) \to (x = y).$$

Corollary 10.2.5. *Any injective map into a set is an embedding.*

Proof. Let $f : A \to B$ be an injective map between sets. Now consider the relation

$$R(x,y) :\equiv (f(x) = f(y)).$$

Note that R is reflexive, and that $R(x,y)$ is a proposition for each $x, y : A$. Moreover, by the assumption that f is injective, we have

$$R(x,y) \to (x = y)$$

for any $x, y : A$. Therefore we are in the situation of Theorem 10.2.3, so it follows that the map $\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$ is an equivalence. □

Theorem 10.2.6. *The type of natural numbers is a set.*

Proof. We will apply Theorem 10.2.3. Note that the observational equality $\mathsf{Eq}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ on $\mathbb{N}$ (Definition 6.4.2) is a reflexive relation by Exercise 6.1, and moreover that $\mathsf{Eq}_{\mathbb{N}}(n,m)$ is a proposition for every $n, m : \mathbb{N}$ (proof by double induction). Therefore it suffices to show that

$$\prod_{(m,n:\mathbb{N})} \mathsf{Eq}_{\mathbb{N}}(m,n) \to (m = n).$$

This follows from the fact that observational equality is the *least* reflexive relation, which was shown in Lemma 6.4.3. □

10.3 General truncation levels

Definition 10.3.1. We define $\mathsf{is\text{-}trunc} : \mathbb{Z}_{\geq -2} \to \mathcal{U} \to \mathcal{U}$ by induction on $k : \mathbb{Z}_{\geq -2}$, taking

$$\begin{aligned}\mathsf{is\text{-}trunc}_{-2}(A) &:\equiv \mathsf{is\text{-}contr}(A) \\ \mathsf{is\text{-}trunc}_{k+1}(A) &:\equiv \prod_{(x,y:A)} \mathsf{is\text{-}trunc}_k(x = y).\end{aligned}$$

For any type A, we say that A is k**-truncated**, or a k**-type**, if there is a term of type $\mathsf{is\text{-}trunc}_k(A)$. We say that a map $f : A \to B$ is k-truncated if its fibers are k-truncated.

Theorem 10.3.2. *If A is a k-type, then A is also a $(k+1)$-type.*

Proof. We have seen in Example 10.1.2 that contractible types are propositions. This proves the base case. For the inductive step, note that if any k-type is also a $(k+1)$-type, then any $(k+1)$-type is a $(k+2)$-type, since its identity types are k-types and therefore $(k+1)$-types. □

Theorem 10.3.3. *If $e : A \simeq B$ is an equivalence, and B is a k-type, then so is A.*

Proof. We have seen in Exercise 8.3 that if B is contractible and $e : A \simeq B$ is an equivalence, then A is also contractible. This proves the base case.

For the inductive step, assume that the k-types are stable under equivalences, and consider $e : A \simeq B$ where B is a $(k+1)$-type. In Theorem 9.3.2 we have seen that

$$\mathsf{ap}_e : (x = y) \to (e(x) = e(y))$$

is an equivalence for any x, y. Note that $e(x) = e(y)$ is a k-type, so by the induction hypothesis it follows that $x = y$ is a k-type. This proves that A is a $(k+1)$-type. □

Corollary 10.3.4. *If $f : A \to B$ is an embedding, and B is a $(k+1)$-type, then so is A.*

Proof. By the assumption that f is an embedding, the action on paths

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

is an equivalence for every $x, y : A$. Since B is assumed to be a $(k+1)$-type, it follows that $f(x) = f(y)$ is a k-type for every $x, y : A$. Therefore we conclude by Theorem 10.3.3 that $x = y$ is a k-type for every $x, y : A$. In other words, A is a $(k+1)$-type. □

Theorem 10.3.5. *Let B be a type family over A. Then the following are equivalent:*

*(i) For each $x : A$ the type $B(x)$ is k-truncated. In this case we say that the family B is **k-truncated**.*

(ii) The projection map

$$\mathsf{pr}_1 : \left(\textstyle\sum_{(x:A)} B(x)\right) \to A$$

is k-truncated.

Proof. By Exercise 8.6 we obtain equivalences

$$\mathsf{fib}_{\mathsf{pr}_1}(x) \simeq B(x)$$

for every $x : A$. Therefore the claim follows from Theorem 10.3.3. □

Theorem 10.3.6. *Let $f : A \to B$ be a map. The following are equivalent:*

(i) The map f is $(k+1)$-truncated.

(ii) For each $x, y : A$, the map

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

is k-truncated.

Proof. First we show that for any $s, t : \mathsf{fib}_f(b)$ there is an equivalence

$$(s = t) \simeq \mathsf{fib}_{\mathsf{ap}_f}(\mathsf{pr}_2(s) \cdot \mathsf{pr}_2(t)^{-1})$$

We do this by Σ-induction on s and t, and then we calculate

$$\begin{aligned}
((x,p) = (y,q)) &\simeq \mathsf{Eq\text{-}fib}_f((x,p),(y,q)) \\
&\equiv \textstyle\sum_{(\alpha:x=y)} p = \mathsf{ap}_f(\alpha) \cdot q \\
&\simeq \textstyle\sum_{(\alpha:x=y)} \mathsf{ap}_f(\alpha) \cdot q = p \\
&\simeq \textstyle\sum_{(\alpha:x=y)} \mathsf{ap}_f(\alpha) = p \cdot q^{-1} \\
&\equiv \mathsf{fib}_{\mathsf{ap}_f}(p \cdot q^{-1}).
\end{aligned}$$

By these equivalences, it follows that if ap_f is k-truncated, then for each $s, t : \mathsf{fib}_f(b)$ the identity type $s = t$ is equivalent to a k-truncated type, and therefore we obtain by Theorem 10.3.3 that f is $(k+1)$-truncated.

For the converse, note that we have equivalences

$$\mathsf{fib}_{\mathsf{ap}_f}(p) \simeq ((x,p) = (y, \mathsf{refl}_{f(y)})).$$

It follows that if f is $(k+1)$-truncated, then the identity type $(x,p) = (y, \mathsf{refl}_{f(y)})$ in $\mathsf{fib}_f(f(y))$ is k-truncated for any $p : f(x) = f(y)$. We conclude by Theorem 10.3.3 that the fiber $\mathsf{fib}_{\mathsf{ap}_f}(p)$ is k-truncated. □

Corollary 10.3.7. *A map is an embedding if and only if its fibers are propositions.*

Corollary 10.3.8. *A type family B over A is a subtype if and only if the projection map*

$$\mathsf{pr}_1 : \left(\textstyle\sum_{(x:A)} B(x)\right) \to A$$

is an embedding.

Theorem 10.3.9. *Let $f : \prod_{(x:A)} B(x) \to C(x)$ be a family of maps. Then the following are equivalent:*

(i) For each $x : A$ the map $f(x)$ is k-truncated.

(ii) The induced map

$$\mathsf{tot}(f) : \left(\textstyle\sum_{(x:A)} B(x)\right) \to \left(\textstyle\sum_{(x:A)} C(x)\right)$$

is k-truncated.

Proof. This follows directly from Lemma 9.1.2 and Theorem 10.3.3. □

Exercises

10.1 (a) Show that $\mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ is an embedding.
(b) Show that $n \mapsto m + n$ is an embedding, for each $m : \mathbb{N}$. Moreover, conclude that there is an equivalence

$$\mathsf{fib}_{\mathsf{add}_{\mathbb{N}}(m)}(n) \simeq (m \leq n).$$

(c) Show that $n \mapsto mn$ is an embedding, for each $m > 0$ in $\mathbb{N}$. Conclude that the divisibility relation

$$d \mid n$$

is a proposition for each $d, n : \mathbb{N}$ such that $d > 0$.

10.2 Let A be a type, and let the **diagonal** of A be the map $\delta_A : A \to A \times A$ given by $\lambda x.\,(x, x)$.

(a) Show that

$$\mathsf{is\text{-}equiv}(\delta_A) \leftrightarrow \mathsf{is\text{-}prop}(A).$$

(b) Construct an equivalence $\mathsf{fib}_{\delta_A}((x,y)) \simeq (x = y)$ for any $x, y : A$.

(c) Show that A is $(k+1)$-truncated if and only if $\delta_A : A \to A \times A$ is k-truncated.

10.3 (a) Let B be a type family over A. Show that if A is a k-type, and $B(x)$ is a k-type for each $x : A$, then so is $\sum_{(x:A)} B(x)$. Conclude that for any two k-types A and B, the type $A \times B$ is also a k-type. Hint: for the base case, use Exercises 8.3 and 8.5.

(b) Show that for any k-type A, the identity types of A are also k-types.

(c) Show that any maps $f : A \to B$ between k-types A and B is a k-truncated map.

(d) Use Exercise 8.6 to show that for any type family $B : A \to \mathcal{U}$, if A and $\sum_{(x:A)} B(x)$ are k-types, then so is $B(x)$ for each $x : A$.

10.4 Show that $\mathbf{2}$ is a set by applying Theorem 10.2.3 with the observational equality on $\mathbf{2}$ defined in Exercise 6.4.

10.5 Show that for any two $(k+2)$-types A and B, the disjoint sum $A + B$ is again a $(k+2)$-type. Conclude that $\mathbb{Z}$ is a set.

10.6 Use Exercises 8.2 and 7.8 to show that if A is a retract of a k-type B, then A is also a k-type.

10.7 Show that a type A is a $(k+1)$-type if and only if the map $\mathsf{const}_x : \mathbf{1} \to A$ is k-truncated for every $x : A$.

10.8 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$, and suppose that g is k-truncated. Show that f is k-truncated if and only if h is k-truncated.

11 Function extensionality

11.1 Equivalent forms of function extensionality

Axiom 11.1.1 (Function Extensionality)**.** *For any type family B over A, and any two dependent functions $f, g : \prod_{(x:A)} B(x)$, the canonical map*

$$\mathsf{htpy\text{-}eq} : (f = g) \to (f \sim g)$$

that sends refl_f to $\mathsf{refl\text{-}htpy}_f$ is an equivalence. We will write $\mathsf{eq\text{-}htpy}$ for its inverse.

In other words, the axiom of function extensionality asserts that for any two dependent functions $f, g : \prod_{(x:A)} B(x)$, the type of identifications $f = g$ is equivalent to the type of homotopies $f \sim g$ from f to g. By the fundamental theorem of identity types (Theorem 9.2.2) there are three equivalent ways of asserting function extensionality. In the following theorem we state one further equivalent condition.

Theorem 11.1.2. *The following are equivalent:*

(i) *The axiom of function extensionality.*

(ii) *For any type family B over A and any dependent function $f : \prod_{(x:A)} B(x)$, the total space*

$$\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$$

is contractible.

(iii) *The principle of **homotopy induction**: for any type family B over A, any dependent function $f : \prod_{(x:A)} B(x)$, and any family of types $P(g, H)$ indexed by $g : \prod_{(x:A)} B(x)$ and $H : f \sim g$, the evaluation function*

$$\Big(\prod_{(g:\prod_{(x:A)} B(x))} \prod_{(H:f\sim g)} P(g, H)\Big) \to P(f, \mathsf{refl\text{-}htpy}_f)$$

given by $s \mapsto s(f, \mathsf{refl\text{-}htpy}_f)$ has a section.

(iv) *The **weak function extensionality principle** holds: For every type family B over A one has*

$$\Big(\prod_{(x:A)} \mathsf{is\text{-}contr}(B(x))\Big) \to \mathsf{is\text{-}contr}\Big(\prod_{(x:A)} B(x)\Big).$$

Proof. The fact that function extensionality is equivalent to (ii) and (iii) follows directly from Theorem 9.2.2.

To show that function extensionality implies weak function extensionality, suppose that each $B(a)$ is contractible with center of contraction $c(a)$ and contraction $C_a : \prod_{(y:B(a))} c(a) = y$. Then we take $c :\equiv \lambda a.\, c(a)$ to be the center of contraction of $\prod_{(x:A)} B(x)$. To construct the contraction we have to define a term of type

$$\prod_{(f:\prod_{(x:A)} B(x))} c = f.$$

Let $f : \prod_{(x:A)} B(x)$. By function extensionality we have a map $(c \sim f) \to (c = f)$, so it suffices to construct a term of type $c \sim f$. Here we take $\lambda a.\, C_a(f(a))$. This completes the proof that function extensionality implies weak function extensionality.

In the remaining part of the proof, we will show that weak function extensionality implies that the type

$$\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$$

is contractible for any $f : \prod_{(x:A)} B(x)$. In order to do this, we first note that we have a section-retraction pair

$$\Big(\sum_{(g:\prod_{(x:A)} B(x))} f \sim g\Big) \xrightarrow{i} \Big(\prod_{(x:A)} \sum_{(b:B(x))} f(x) = b\Big) \xrightarrow{r} \Big(\sum_{(g:\prod_{(x:A)} B(x))} f \sim g\Big).$$

Here we have the functions

$$i :\equiv \lambda(g, H).\, \lambda x.\, (g(x), H(x))$$
$$r :\equiv \lambda p.\, (\lambda x.\, \mathsf{pr}_1(p(x)), \lambda x.\, \mathsf{pr}_2(p(x))).$$

Their composite is homotopic to the identity function by the computation rule for Σ-types and the η-rule for Π-types:

$$r(i(g, H)) \equiv r(\lambda x.\, (g(x), H(x)))$$

$$\begin{aligned} &\equiv (\lambda x.\, g(x), \lambda x.\, H(x)) \\ &\equiv (g, H). \end{aligned}$$

Now we observe that the type $\prod_{(x:A)} \sum_{(b:B(x))} f(x) = b$ is a product of contractible types, so it is contractible by our assumption of the weak function extensionality principle. The claim therefore follows, since retracts of contractible types are contractible by Exercise 8.2. □

For the remainder of this chapter we will assume that the function extensionality axiom holds. In Theorem 12.2.2 we will derive function extensionality from the univalence axiom.

As a first application of the function extensionality axiom we generalize the weak function extensionality axiom to k-types.

Theorem 11.1.3. *For any type family B over A one has*

$$\Big(\prod_{(x:A)} \mathsf{is\text{-}trunc}_k(B(x))\Big) \to \mathsf{is\text{-}trunc}_k\Big(\prod_{(x:A)} B(x)\Big).$$

Proof. The theorem is proven by induction on $k \geq -2$. The base case is just the weak function extensionality principle, which was shown to follow from function extensionality in Theorem 11.1.2.

For the inductive hypothesis, assume that the k-types are closed under dependent function types. Assume that B is a family of $(k+1)$-types. By function extensionality, the type $f = g$ is equivalent to $f \sim g$ for any two dependent functions $f, g : \prod_{(x:A)} B(x)$. Now observe that $f \sim g$ is a dependent product of k-types, and therefore it is an k-type by our inductive hypotheses. Therefore, it follows by Theorem 10.3.3 that $f = g$ is an k-type, and hence that $\prod_{(x:A)} B(x)$ is an $(k+1)$-type. □

Corollary 11.1.4. *Suppose B is a k-type. Then $A \to B$ is also a k-type, for any type A.*

11.2 The type theoretic principle of choice

The type theoretic principle of choice asserts that Π distributes over Σ. More precisely, it asserts that the canonical map

$$\mathsf{choice} : \Big(\prod_{(x:A)} \sum_{(y:B(x))} C(x,y)\Big) \to \Big(\sum_{(f:\prod_{(x:A)} B(x))} \prod_{(x:A)} C(x, f(x))\Big)$$

given by $\lambda h.\, (\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x)))$, is an equivalence. In order to see this as a principle of choice, one can view the left hand side as the type of functions h that pick for every $x : A$ a term $y : B(x)$ equipped with a term of type $C(x,y)$. The function choice then constructs a dependent function $f : \prod_{(x:A)} B(x)$ equipped with a term of type $\prod_{(x:A)} C(x, f(x))$. In this section we show that the map choice is an equivalence, and we use this to characterize the identity of any dependent function type $\prod_{(x:A)} B(x)$ in terms of any characterization of the identity types of the individual types $B(x)$.

Theorem 11.2.1. *Consider a family of types $C(x,y)$ indexed by $x : A$ and $y : B(x)$. Then the map*

$$\mathsf{choice} : \Big(\prod_{(x:A)} \sum_{(y:B(x))} C(x,y)\Big) \to \Big(\sum_{(f:\prod_{(x:A)} B(x))} \prod_{(x:A)} C(x, f(x))\Big)$$

given by $\lambda h.\, (\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x)))$ is an equivalence.

Proof. We define the map

$$\mathsf{choice}^{-1} : \Big(\sum_{(f:\prod_{(x:A)} B(x))} \prod_{(x:A)} C(x, f(x))\Big) \to \Big(\prod_{(x:A)} \sum_{(y:B(x))} C(x,y)\Big)$$

by $\lambda(f,g).\,\lambda x.\,(f(x), g(x))$. Then we have to construct homotopies

$$\mathsf{choice} \circ \mathsf{choice}^{-1} \sim \mathsf{id}, \qquad \text{and} \qquad \mathsf{choice}^{-1} \circ \mathsf{choice} \sim \mathsf{id}.$$

For the first homotopy it suffices to construct an identification

$$\mathsf{choice}(\mathsf{choice}^{-1}(f,g)) = (f,g)$$

for any $f : \prod_{(x:A)} B(x)$ and any $g : \prod_{(x:A)} C(x, f(x))$. We compute the left-hand side as follows:

$$\begin{aligned} \mathsf{choice}(\mathsf{choice}^{-1}(f,g)) &\equiv \mathsf{choice}(\lambda x.\,(f(x), g(x))) \\ &\equiv (\lambda x.\,f(x), \lambda x.\,g(x)). \end{aligned}$$

By the η-rule it follows that $f \equiv \lambda x.\,f(x)$ and $g \equiv \lambda x.\,g(x)$. Therefore we have the identification

$$\mathsf{refl}_{(f,g)} : \mathsf{choice}(\mathsf{choice}^{-1}(f,g)) = (f,g).$$

This completes the construction of the first homotopy.

For the second homotopy we have to construct an identification

$$\mathsf{choice}^{-1}(\mathsf{choice}(h)) = h$$

for any $h : \prod_{(x:A)} \sum_{(y:B(x))} C(x,y)$. We compute the left-hand side as follows:

$$\begin{aligned} \mathsf{choice}^{-1}(\mathsf{choice}(h)) &\equiv \mathsf{choice}^{-1}(\lambda x.\,\mathsf{pr}_1(h(x)), (\lambda x.\,\mathsf{pr}_2(h(x)))) \\ &\equiv \lambda x.\,(\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x))) \end{aligned}$$

However, it is *not* the case that $(\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x))) \equiv h(x)$ for any $h : \prod_{(x:A)} \sum_{(y:B(x))} C(x,y)$. Nevertheless, we have the identification

$$\mathsf{eq\text{-}pair}(\mathsf{refl}, \mathsf{refl}) : (\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x))) = h(x).$$

Therefore we obtain the required homotopy by function extensionality:

$$\lambda h.\,\mathsf{eq\text{-}htpy}(\lambda x.\,\mathsf{eq\text{-}pair}(\mathsf{refl}_{\mathsf{pr}_1(h(x))}, \mathsf{refl}_{\mathsf{pr}_2(h(x))})) : \mathsf{choice}^{-1} \circ \mathsf{choice} \sim \mathsf{id}. \qquad \square$$

Corollary 11.2.2. *For type A and any type family C over B, the map*

$$\Big(\sum_{(f:A\to B)} \prod_{(x:A)} C(f(x))\Big) \to \Big(A \to \sum_{(y:B)} C(x)\Big)$$

given by $\lambda(f,g).\,\lambda x.\,(f(x), g(x))$ is an equivalence.

Remark 11.2.3. The type theoretic choice principle can be used to derive the binomial theorem. We give an informal argument of how this goes. Recall that the binomial theorem asserts that

$$(n+m)^k = \sum_{l=0}^{k} \binom{k}{l} n^l m^{k-l}$$

for any three natural numbers k, m, n.

Consider the types $A :\equiv \mathsf{Fin}(k)$, $B :\equiv \mathsf{Fin}(n)$ and $C :\equiv \mathsf{Fin}(m)$. Then we can define the type family $P : \mathbf{2} \to \mathcal{U}$ given by

$$\begin{aligned} P(1_{\mathbf{2}}) &:\equiv B \\ P(0_{\mathbf{2}}) &:\equiv C. \end{aligned}$$

Now, the type theoretic principle of choice gives us an equivalence

$$\Big(\prod_{(x:A)}\sum_{(t:\mathbf{2})} P(t)\Big) \simeq \Big(\sum_{(f:A\to\mathbf{2})}\prod_{(x:A)} P(f(x))\Big).$$

Now we note that the type $(f(x) = 1) + (f(x) = 0)$ is contractible for any $f : A \to \mathbf{2}$ and $x : A$. Therefore we have equivalences

$$\begin{aligned} \sum_{(f:A\to\mathbf{2})}\prod_{(x:A)} P(f(x)) &\simeq \sum_{(f:A\to\mathbf{2})}\prod_{(x:A)}\prod_{(t:(f(x)=1)+(f(x)=0))} P(f(x)) \\ &\simeq \sum_{(f:A\to\mathbf{2})} (\mathsf{fib}_f(1) \to B) \times (\mathsf{fib}_f(0) \to C) \end{aligned}$$

Now we note that, because there are $\binom{k}{l}$ ways to choose a subset of l elements of A, there are

$$\sum_{l=0}^{k} \binom{k}{l} n^l m^{k-l}$$

elements in the above type.

11.3 Universal properties

The function extensionality principle allows us to prove *universal properties*. Universal properties are characterizations of all maps out of or into a given type, so they are very important. Among other applications, universal properties characterize a type up to equivalence. In the following theorem we prove the universal property of dependent pair types.

Theorem 11.3.1. *Let B be a type family over A, and let X be a type. Then the map*

$$\mathsf{ev\text{-}pair} : \Big(\Big(\sum_{(x:A)} B(x)\Big) \to X\Big) \to \Big(\prod_{(x:A)} (B(x) \to X)\Big)$$

given by $f \mapsto \lambda a.\, \lambda b.\, f(a, b)$ is an equivalence.

Proof. The map in the converse direction is simply

$$\mathsf{ind}_{\Sigma} : \Big(\prod_{(x:A)} (B(x) \to X)\Big) \to \Big(\Big(\sum_{(x:A)} B(x)\Big) \to X\Big).$$

By the computation rules for Σ-types we have

$$\lambda f.\, \mathsf{refl}_f : \mathsf{ev\text{-}pair} \circ \mathsf{ind}_{\Sigma} \sim \mathsf{id}$$

To show that $\mathsf{ind}_{\Sigma} \circ \mathsf{ev\text{-}pair} \sim \mathsf{id}$ we will also apply function extensionality. Thus, it suffices to show that $\mathsf{ind}_{\Sigma}(\lambda x.\, \lambda y.\, f((x, y))) = f$. We apply function extensionality again, so it suffices to show that

$$\prod_{(t:\sum_{(x:A)} B(x))} \mathsf{ind}_{\Sigma}\big(\lambda x.\, \lambda y.\, f((x, y))\big)(t) = f(t).$$

We obtain this homotopy by another application of Σ-induction. $\square$

Corollary 11.3.2. *Let A, B, and X be types. Then the map*

$$\mathsf{ev\text{-}pair} : (A \times B \to X) \to (A \to (B \to X))$$

given by $f \mapsto \lambda a.\, \lambda b.\, f((a,b))$ is an equivalence.

The universal property of identity types is sometimes called the *type theoretical Yoneda lemma*: families of maps out of the identity type are uniquely determined by their action on the reflexivity identification.

Theorem 11.3.3. *Let B be a type family over A, and let $a : A$. Then the map*

$$\mathsf{ev\text{-}refl} : \Big(\prod_{(x:A)}(a = x) \to B(x)\Big) \to B(a)$$

given by $\lambda f.\, f(a, \mathsf{refl}_a)$ is an equivalence.

Proof. The inverse φ is defined by path induction, taking $b : B(a)$ to the function f satisfying $f(a, \mathsf{refl}_a) \equiv b$. It is immediate that $\mathsf{ev\text{-}refl} \circ \varphi \sim \mathsf{id}$.

To see that $\varphi \circ \mathsf{ev\text{-}refl} \sim \mathsf{id}$, let $f : \prod_{(x:A)}(a = x) \to B(x)$. To show that $\varphi(f(a, \mathsf{refl}_a)) = f$ we use function extensionality (twice), so it suffices to show that

$$\prod_{(x:A)}\prod_{(p:a=x)}\varphi(f(a, \mathsf{refl}_a), x, p) = f(x, p).$$

This follows by path induction on p, since $\varphi(f(a, \mathsf{refl}_a), a, \mathsf{refl}_a) \equiv f(a, \mathsf{refl}_a)$. □

11.4 Composing with equivalences

We show in this section that a map $f : A \to B$ is an equivalence if and only if for any type X the precomposition map

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

is an equivalence. Moreover, we will show in Theorem 11.4.1 that the 'dependent version' of this statement also holds: a map $f : A \to B$ is an equivalence if and only if for any type family P over B, the precomposition map

$$- \circ f : \Big(\prod_{(y:B)}P(y)\Big) \to \Big(\prod_{(x:A)}P(f(x))\Big)$$

is an equivalence.

Theorem 11.4.1. *For any map $f : A \to B$, the following are equivalent:*

(i) f is an equivalence.

(ii) For any type family P over B the map

$$\Big(\prod_{(y:B)}P(y)\Big) \to \Big(\prod_{(x:A)}P(f(x))\Big)$$

given by $h \mapsto h \circ f$ is an equivalence.

(iii) For any type X the map

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

given by $g \mapsto g \circ f$ is an equivalence.

Proof. To show that (i) implies (ii), we first recall from Lemma 8.3.5 that any equivalence is also coherently invertible. Therefore f comes equipped with

$$\begin{aligned} g &: B \to A \\ G &: f \circ g \sim \mathsf{id}_B \\ H &: g \circ f \sim \mathsf{id}_A \\ K &: G \cdot f \sim f \cdot H. \end{aligned}$$

Then we define the inverse of $- \circ f$ to be the map

$$\varphi : \Big(\prod_{(x:A)} P(f(x))\Big) \to \Big(\prod_{(y:B)} P(y)\Big)$$

given by $h \mapsto \lambda y.\, \mathsf{tr}_P(G(y), h(g(y)))$.

To see that φ is a section of $- \circ f$, let $h : \prod_{(x:A)} P(f(x))$. By function extensionality it suffices to construct a homotopy $\varphi(h) \circ f \sim h$. In other words, we have to show that

$$\mathsf{tr}_P(G(f(x)), h(g(f(x))) = h(x)$$

for any $x : A$. Now we use the additional homotopy K from our assumption that f is coherently invertible. Since we have $K(x) : G(f(x)) = \mathsf{ap}_f(H(x))$ it suffices to show that

$$\mathsf{tr}_P(\mathsf{ap}_f(H(x)), hgf(x)) = h(x).$$

A simple path-induction argument yields that

$$\mathsf{tr}_P(\mathsf{ap}_f(p)) \sim \mathsf{tr}_{P \circ f}(p)$$

for any path $p : x = y$ in A, so it suffices to construct an identification

$$\mathsf{tr}_{P \circ f}(H(x), hgf(x)) = h(x).$$

We have such an identification by $\mathsf{apd}_h(H(x))$.

To see that φ is a retraction of $- \circ f$, let $h : \prod_{(y:B)} P(y)$. By function extensionality it suffices to construct a homotopy $\varphi(h \circ f) \sim h$. In other words, we have to show that

$$\mathsf{tr}_P(G(y), hfg(y)) = h(y)$$

for any $y : B$. We have such an identification by $\mathsf{apd}_h(G(y))$. This completes the proof that (i) implies (ii).

Note that (iii) is an immediate consequence of (ii), since we can just choose P to be the constant family X.

It remains to show that (iii) implies (i). Suppose that

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

is an equivalence for every type X. Then its fibers are contractible by Theorem 8.3.6. In particular, choosing $X \equiv A$ we see that the fiber

$$\mathsf{fib}_{- \circ f}(\mathsf{id}_A) \equiv \sum_{(h:B \to A)} h \circ f = \mathsf{id}_A$$

is contractible. Thus we obtain a function $h : B \to A$ and a homotopy $H : h \circ f \sim \mathsf{id}_A$ showing that h is a retraction of f. We will show that h is also a section of f. To see this, we use that the fiber

$$\mathsf{fib}_{-\circ f}(f) \equiv \textstyle\sum_{(i:B\to B)} i \circ f = f$$

is contractible (choosing $X \equiv B$). Of course we have $(\mathsf{id}_B, \mathsf{refl}_f)$ in this fiber. However we claim that there also is an identification $p : (f \circ h) \circ f = f$, showing that $(f \circ h, p)$ is in this fiber, because

$$\begin{aligned}(f \circ h) \circ f &\equiv f \circ (h \circ f)\\ &= f \circ \mathsf{id}_A\\ &\equiv f\end{aligned}$$

Now we conclude by the contractibility of the fiber that $(\mathsf{id}_B, \mathsf{refl}_f) = (f \circ h, p)$. In particular we obtain that $\mathsf{id}_B = f \circ h$, showing that h is a section of f. □

Exercises

11.1 Show that the functions

$$\begin{aligned}\mathsf{inv\text{-}htpy} &: (f \sim g) \to (g \sim f)\\ \mathsf{concat\text{-}htpy}(H) &: (g \sim h) \to (f \sim h)\\ \mathsf{concat\text{-}htpy}'(K) &: (f \sim g) \to (f \sim h)\end{aligned}$$

are equivalences for every $f, g, h : \prod_{(x:A)} B(x)$. Here, $\mathsf{concat\text{-}htpy}'(K)$ is the function defined by $H \mapsto H \cdot K$.

11.2 (a) Show that for any type A the type $\mathsf{is\text{-}contr}(A)$ is a proposition.
(b) Show that for any type A and any $k \geq -2$, the type $\mathsf{is\text{-}trunc}_k(A)$ is a proposition.

11.3 Let $f : X \to Y$ be a map. Show that the following are equivalent:
(i) f is an equivalence.
(ii) The map $f \circ - : X^A \to Y^A$ is an equivalence for every type A.

11.4 Let $f : A \to B$ be a function.
(a) Show that if f is an equivalence, then the type $\sum_{(g:B\to A)} f \circ g \sim \mathsf{id}$ of sections of f is contractible.
(b) Show that if f is an equivalence, then the type $\sum_{(h:B\to A)} h \circ f \sim \mathsf{id}$ of retractions of f is contractible.
(c) Show that $\mathsf{is\text{-}equiv}(f)$ is a proposition.
(d) Use Exercises 11.2 and 11.5 to show that $\mathsf{is\text{-}equiv}(f) \simeq \mathsf{is\text{-}contr}(f)$.

Conclude that $A \simeq B$ is a subtype of $A \to B$, and in particular that the map $\mathsf{pr}_1 : (A \simeq B) \to (A \to B)$ is an embedding.

11.5 (a) Let P and Q be propositions. Show that

$$(P \leftrightarrow Q) \simeq (P \simeq Q).$$

(b) Show that P is a proposition if and only if $P \to P$ is contractible.

11.6 Show that $\mathsf{path\text{-}split}(f)$ and $\mathsf{is\text{-}coh\text{-}invertible}(f)$ are propositions for any map $f : A \to B$. Conclude that we have equivalences

$$\mathsf{is\text{-}equiv}(f) \simeq \mathsf{path\text{-}split}(f) \simeq \mathsf{is\text{-}coh\text{-}invertible}(f).$$

11.7 Construct for any type A an equivalence

$$\mathsf{has\text{-}inverse}(\mathsf{id}_A) \simeq \Big(\mathsf{id}_A \sim \mathsf{id}_A\Big).$$

Note: We will use this fact in Exercise 19.6 to show that there are types for which $\mathsf{is\text{-}invertible}(\mathsf{id}_A) \not\simeq \mathsf{is\text{-}equiv}(\mathsf{id}_A)$.

11.8 (a) Show that the type

$$\prod_{(t:\emptyset)} P(t)$$

is contractible for any $P : \emptyset \to \mathcal{U}$.

(b) Show that for any type X the following are equivalent:

(i) the unique map $\emptyset \to X$ is an equivalence.

(ii) The type Y^X is contractible for any type Y.

11.9 Consider two types A and B.

(a) Show that the map

$$\mathsf{ev\text{-}inl\text{-}inr} : \Big(\prod_{(t:A+B)} P(t)\Big) \to \Big(\prod_{(x:A)} P(\mathsf{inl}(x))\Big) \times \Big(\prod_{(y:B)} P(\mathsf{inr}(y))\Big)$$

given by $f \mapsto (f \circ \mathsf{inl}, f \circ \mathsf{inr})$ is an equivalence.

(b) Show that the following are equivalent for any type X equipped with maps $i : A \to X$ and $j : B \to X$:

(i) The map $\mathsf{ind}_+(i, j) : A + B \to X$ is an equivalence.

(ii) For any type Y, the map

$$\lambda f.\, (f \circ i, f \circ j) : (X \to Y) \to (A \to Y) \times (B \to Y)$$

is an equivalence.

11.10 (a) Show that the map

$$\Big(\prod_{(t:\mathbf{1})} P(t)\Big) \to P(\star)$$

given by $\lambda f.\, f(\star)$ is an equivalence.

(b) Consider a type X equipped with a point $x : X$. Show that the following are equivalent:

(i) The map $\mathsf{ind}_{\mathbf{1}}(x) : \mathbf{1} \to X$ is an equivalence (i.e., X is contractible).

(ii) For any type Y the map

$$\lambda f.\, f(x) : (X \to Y) \to Y$$

is an equivalence.

11.11 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$.

(a) Show that if h has a section, then $\mathsf{sec}(g)$ is a retract of $\mathsf{sec}(f)$.

(b) Show that if g has a retraction, then $\mathsf{retr}(h)$ is a retract of $\mathsf{sec}(f)$.

11.12 Consider a family $f_i : A_i \to B_i$ of k-truncated maps, indexed by $i : I$. Show that the map

$$\lambda h.\, \lambda i.\, f_i(h(i)) : \Big(\prod_{(i:I)} A_i\Big) \to \Big(\prod_{(i:I)} B_i\Big)$$

is again k-truncated. Conclude that if each f_i is an equivalence, then so is the above map.

11.13 Consider a map $f : X \to Y$. Show that the following are equivalent:

(i) The map f is k-truncated.

(ii) For every type A, the postcomposition function

$$f \circ - : (A \to X) \to (A \to Y)$$

is k-truncated.

In particular it follows that f is an embedding if and only if $f \circ -$ is an embedding.
Hint: Show that the square

$$\begin{tikzcd} (f = g) \arrow[r, "\mathsf{ap}_m"] \arrow[d, "\mathsf{htpy\text{-}eq}"'] & (m \circ f = m \circ g) \arrow[d, "\mathsf{htpy\text{-}eq}"] \\ (f \sim g) \arrow[r, "H \mapsto m \cdot H"'] & (m \circ f \sim m \circ g) \end{tikzcd}$$

commutes, and apply Exercise 11.12.

11.14 Consider a function $f : A \to B$, and let P be a family of types over B. Show that the map

$$\Big(\prod_{(b:B)} \mathsf{fib}_f(b) \to P(b)\Big) \to \Big(\prod_{(a:A)} P(f(a))\Big)$$

given by $h \mapsto h_{f(a)}(a, \mathsf{refl}_{f(a)})$ is an equivalence.

11.15 Consider a diagram of the form

$$\begin{tikzcd} A \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

(a) Show that the type $\sum_{(h:A\to B)} f \sim g \circ h$ is equivalent to the type of families of maps

$$\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(x).$$

(b) Show that the type $\sum_{(h:A\simeq B)} f \sim g \circ h$ is equivalent to the type of families of equivalences

$$\prod_{(x:X)} \mathsf{fib}_f(x) \simeq \mathsf{fib}_g(x).$$

11.16 Consider a diagram of the form

$$\begin{tikzcd} A \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r, "h"'] & Y. \end{tikzcd}$$

Show that the type $\sum_{(i:A\to B)} h \circ f \sim g \circ i$ is equivalent to the type of families of maps

$$\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(h(x)).$$

11.17 Let A and B be sets. Show that type type $A \simeq B$ of equivalences from A to B is equivalent to the type $A \cong B$ of **isomorphisms** from A to B, i.e., the type of quadruples (f, g, H, K) consisting of

$$\begin{aligned} f &: A \to B \\ g &: B \to A \\ H &: f \circ g = \mathsf{id}_B \\ K &: g \circ f = \mathsf{id}_A. \end{aligned}$$

11.18 Let B be a type family over A, and consider the postcomposition function

$$\mathsf{pr}_1 \circ - : \Big(\textstyle\sum_{(x:A)} B(x)\Big)^A \to A^A.$$

Construct equivalences

$$\Big(\prod_{(x:A)} B(x)\Big) \simeq \mathsf{sec}(\mathsf{pr}_1) \simeq \mathsf{fib}_{\mathsf{pr}_1 \circ -}(\mathsf{id}_A).$$

11.19 Construct equivalences

$$\begin{aligned} \mathsf{Fin}(n^m) &\simeq (\mathsf{Fin}(m) \to \mathsf{Fin}(n)) \\ \mathsf{Fin}(n!) &\simeq (\mathsf{Fin}(n) \simeq \mathsf{Fin}(n)). \end{aligned}$$

12 The univalence axiom

12.1 Equivalent forms of the univalence axiom

The univalence axiom characterizes the identity type of the universe. Roughly speaking, it asserts that equivalent types are equal. It is considered to be an *extensionality principle* for types.

Axiom 12.1.1 (Univalence)**.** *The **univalence axiom** on a universe $\mathcal{U}$ is the statement that for any $A : \mathcal{U}$ the family of maps*

$$\mathsf{equiv\text{-}eq} : \prod_{(B:\mathcal{U})} (A = B) \to (A \simeq B).$$

*that sends refl_A to the identity equivalence $\mathsf{id} : A \simeq A$ is a family of equivalences. A universe satisfying the univalence axiom is referred to as a **univalent universe**. If $\mathcal{U}$ is a univalent universe we will write* $\mathsf{eq\text{-}equiv}$ *for the inverse of* $\mathsf{equiv\text{-}eq}$.

The following theorem is a special case of the fundamental theorem of identity types (Theorem 9.2.2). Subsequently we will assume that any type is contained in a univalent universe.

Theorem 12.1.2. *The following are equivalent:*

(i) The univalence axiom holds.

(ii) The type

$$\textstyle\sum_{(B:\mathcal{U})} A \simeq B$$

is contractible for each $A : \mathcal{U}$.

(iii) *The principle of* ***equivalence induction*** *holds: for every $A : \mathcal{U}$ and for every type family*

$$P : \prod_{(B:\mathcal{U})}(A \simeq B) \to \mathcal{U},$$

the map

$$\Big(\prod_{(B:\mathcal{U})}\prod_{(e:A\simeq B)}P(B,e)\Big) \to P(A,\mathsf{id}_A)$$

given by $f \mapsto f(A,\mathsf{id}_A)$ has a section.

12.2 Univalence implies function extensionality

One of the first applications of the univalence axiom was Voevodsky's theorem that the univalence axiom on a universe $\mathcal{U}$ implies function extensionality for types in $\mathcal{U}$. The proof uses the fact that weak function extensionality implies function extensionality.

We will also make use of the following lemma. Note that this statement was also part of Exercise 11.3. That exercise is solved using function extensionality. Since our present goal is to derive function extensionality from the univalence axiom, we cannot make use of that exercise.

Lemma 12.2.1. *For any equivalence $e : X \simeq Y$ in a univalent universe $\mathcal{U}$, and any type A, the post-composition map*

$$e \circ - : (A \to X) \to (A \to Y)$$

is an equivalence.

Proof. The statement is obvious for the identity equivalence $\mathsf{id} : X \simeq X$. Therefore the claim follows by equivalence induction, which is by Theorem 12.1.2 one of the equivalent forms of the univalence axiom. □

Theorem 12.2.2. *For any universe $\mathcal{U}$, the univalence axiom on $\mathcal{U}$ implies function extensionality on $\mathcal{U}$.*

Proof. Note that by Theorem 11.1.2 it suffices to show that univalence implies weak function extensionality, where we note that Theorem 11.1.2 also holds when it is restricted to small types.

Suppose that $B : A \to \mathcal{U}$ is a family of contractible types. Our goal is to show that the product $\prod_{(x:A)}B(x)$ is contractible. Since each $B(x)$ is contractible, the projection map $\mathsf{pr}_1 : \big(\sum_{(x:A)} B(x)\big) \to A$ is an equivalence by Exercise 8.6.

Now it follows by Lemma 12.2.1 that $\mathsf{pr}_1 \circ -$ is an equivalence. Consequently, it follows from Theorem 8.3.6 that the fibers of

$$\mathsf{pr}_1 \circ - : \Big(A \to \textstyle\sum_{(x:A)}B(x)\Big) \to (A \to A)$$

are contractible. In particular, the fiber at id_A is contractible. Therefore it suffices to show that $\prod_{(x:A)}B(x)$ is a retract of $\sum_{(f:A\to\sum_{(x:A)} B(x))} \mathsf{pr}_1 \circ f = \mathsf{id}_A$. In other words, we will construct

$$\Big(\prod_{(x:A)}B(x)\Big) \xrightarrow{i} \Big(\textstyle\sum_{(f:A\to\sum_{(x:A)} B(x))} \mathsf{pr}_1 \circ f = \mathsf{id}_A\Big) \xrightarrow{r} \Big(\prod_{(x:A)}B(x)\Big),$$

and a homotopy $r \circ i \sim \mathsf{id}$.

We define the function i by

$$i(f) :\equiv (\lambda x.\,(x, f(x)), \mathsf{refl}_{\mathsf{id}}).$$

To see that this definition is correct, we need to know that

$$\lambda x.\, \mathsf{pr}_1(x, f(x)) \equiv \mathsf{id}.$$

This is indeed the case, by the η-rule for Π-types.

Next, we define the function r. Let $h : A \to \sum_{(x:A)} B(x)$, and let $p : \mathsf{pr}_1 \circ h = \mathsf{id}$. Then we have the homotopy $H :\equiv \mathsf{htpy\text{-}eq}(p) : \mathsf{pr}_1 \circ h \sim \mathsf{id}$. Then we have $\mathsf{pr}_2(h(x)) : B(\mathsf{pr}_1(h(x)))$ and we have the identification $H(x) : \mathsf{pr}_1(h(x)) = x$. Therefore we define r by

$$r((h, p), x) :\equiv \mathsf{tr}_B(H(x), \mathsf{pr}_2(h(x))).$$

We note that if $p \equiv \mathsf{refl}_{\mathsf{id}}$, then $H(x) \equiv \mathsf{refl}_x$. In this case we have the judgmental equality $r((h, \mathsf{refl}), x) \equiv \mathsf{pr}_2(h(x))$. Thus we see that $r \circ i \equiv \mathsf{id}$ by another application of the η-rule for Π-types. □

12.3 Propositional extensionality and posets

Theorem 12.3.1. *Propositions satisfy* ***propositional extensionality****: for any two propositions P and Q, the canonical map*

$$\mathsf{iff\text{-}eq} : (P = Q) \to (P \leftrightarrow Q)$$

that sends refl_P to $(\mathsf{id}, \mathsf{id})$ is an equivalence. It follows that the type Prop of propositions in $\mathcal{U}$ is a set.

Note that for any $P : \mathsf{Prop}$, we usually also write P for the underlying type of the proposition P. If we would be more formal about it we would have to write $\mathsf{pr}_1(P)$ for the underlying type, since Prop is the Σ-type $\sum_{(X:\mathcal{U})} \mathsf{is\text{-}prop}(X)$. In the following proof it is clearer if we use the more formal notation $\mathsf{pr}_1(P)$ for the underlying type of a proposition P.

Proof. We note that the identity type $P = Q$ is an identity type in Prop. However, since $\mathsf{is\text{-}prop}(X)$ is a proposition for any type X, it follows that the map

$$\mathsf{ap}_{\mathsf{pr}_1} : (P = Q) \to (\mathsf{pr}_1(P) = \mathsf{pr}_1(Q))$$

is an equivalence. Now we observe that we have a commuting square

$$\begin{tikzcd}
(P = Q) \arrow[r] \arrow[d, "\mathsf{ap}_{\mathsf{pr}_1}"'] & (P \leftrightarrow Q) \\
(\mathsf{pr}_1(P) = \mathsf{pr}_1(Q)) \arrow[r, "\mathsf{equiv\text{-}eq}"'] & (\mathsf{pr}_1(P) \simeq \mathsf{pr}_1(Q)) \arrow[u, "\simeq"']
\end{tikzcd}$$

Since the left, bottom, and right map are equivalences, it follows that the top map is an equivalence. □

Definition 12.3.2. A **partially ordered set (poset)** is a set P equipped with a relation

$$- \leq - : P \to (P \to \mathsf{Prop})$$

that is **reflexive** (for every $x : P$ we have $x \leq x$), **transitive** (for every $x, y, z : P$ such that $x \leq y$ and $y \leq z$ we have $x \leq z$), and **anti-symmetric** (for every $x, y : P$ such that $x \leq y$ and $y \leq x$ we have $x = y$).

Remark 12.3.3. The condition that X is a set can be omitted from the definition of a poset. Indeed, if X is any type that comes equipped with a Prop-valued ordering relation $\leq$ that is reflexive and anti-symmetric, then X is a set by Theorem 10.2.3.

Example 12.3.4. The type Prop is a poset, where the ordering relation is given by implication: P is less than Q if $P \to Q$. The fact that $P \to Q$ is a proposition is a special case of Corollary 11.1.4. The relation $P \to Q$ is reflexive by the identity function, and transitive by function composition. Moreover, the relation $P \to Q$ is anti-symmetric by Theorem 12.3.1.

Example 12.3.5. The type of natural numbers comes equipped with at least two important poset structures. The first is given by the usual ordering relation $\leq$, and the second is given by the relation $d \mid n$ that d divides n.

Theorem 12.3.6. *For any poset P and any type X, the set P^X is a poset. In particular the type of subtypes of any type is a poset.*

Proof. Let P be a poset with ordering $\leq$, and let X be a type. Then P^X is a set by Corollary 11.1.4. For any $f, g : X \to P$ we define

$$(f \leq g) :\equiv \prod_{(x:X)} f(x) \leq g(x).$$

Reflexivity and transitivity follow immediately from reflexivity and transitivity of the original relation. Moreover, by the anti-symmetry of the original relation it follows that

$$(f \leq g) \times (g \leq f) \to (f \sim g).$$

Therefore we obtain an identification $f = g$ by function extensionality. The last claim follows immediately from the fact that a subtype of X is a map $X \to$ Prop, and the fact that Prop is a poset. □

Exercises

12.1 (a) Use the univalence axiom to show that the type $\sum_{(A:\mathcal{U})} \mathsf{is\text{-}contr}(A)$ of all contractible types in $\mathcal{U}$ is contractible.
(b) Use Corollaries 10.3.4 and 11.1.4 and Exercise 11.4 to show that if A and B are $(k+1)$-types, then the type $A \simeq B$ is also a $(k+1)$-type.
(c) Use univalence to show that the universe of k-types

$$\mathcal{U}^{\leq k} :\equiv \sum_{(X:\mathcal{U})} \mathsf{is\text{-}trunc}_k(X)$$

is a $(k+1)$-type, for any $k \geq -2$.
(d) Show that $\mathcal{U}^{\leq -1}$ is not a proposition.
(e) Show that $(\mathbf{2} \simeq \mathbf{2}) \simeq \mathbf{2}$, and conclude by the univalence axiom that the universe of sets $\mathcal{U}^{\leq 0}$ is not a set.

12.2 Use the univalence axiom to show that the type $\sum_{(P:\mathsf{Prop})} P$ is contractible.

12.3 Let A and B be small types.

(a) Construct an equivalence

$$(A \to (B \to \mathcal{U})) \simeq \left(\sum_{(S:\mathcal{U})} (S \to A) \times (S \to B)\right)$$

(b) We say that a relation $R : A \to (B \to \mathcal{U})$ is **functional** if it comes equipped with a term of type

$$\mathsf{is\text{-}function}(R) :\equiv \prod_{(x:A)} \mathsf{is\text{-}contr}\Big(\sum_{(y:B)} R(x,y)\Big)$$

For any function $f : A \to B$, show that the **graph** of f

$$\mathsf{graph}_f : A \to (B \to \mathcal{U})$$

given by $\mathsf{graph}_f(a,b) :\equiv (f(a) = b)$ is a functional relation from A to B.

(c) Construct an equivalence

$$\Big(\sum_{(R:A\to(B\to\mathcal{U}))} \mathsf{is\text{-}function}(R)\Big) \simeq (A \to B)$$

(d) Given a relation $R : A \to (B \to \mathcal{U})$ we define the **opposite relation**

$$R^{\mathsf{op}} : B \to (A \to \mathcal{U})$$

by $R^{\mathsf{op}}(y,x) :\equiv R(x,y)$. Construct an equivalence

$$\Big(\sum_{(R:A\to(B\to\mathcal{U}))} \mathsf{is\text{-}function}(R) \times \mathsf{is\text{-}function}(R^{\mathsf{op}})\Big) \simeq (A \simeq B).$$

12.4 (a) Show that $\mathsf{is\text{-}decidable}(P)$ is a proposition, for any proposition P.
(b) Show that $\mathsf{classical\text{-}Prop}$ is equivalent to $\mathbf{2}$.

12.5 Recall that $\mathcal{U}_*$ is the universe of pointed types.

(a) For any (A,a) and (B,b) in $\mathcal{U}_*$, write $(A,a) \simeq_* (B,b)$ for the type of **pointed equivalences** from A to B, i.e.,

$$(A,a) \simeq_* (B,b) :\equiv \sum_{(e:A\simeq B)} e(a) = b.$$

Show that the canonical map

$$((A,a) = (B,b)) \to \Big((A,a) \simeq (B,b)\Big)$$

sending $\mathsf{refl}_{(A,a)}$ to the pair $(\mathsf{id}, \mathsf{refl}_a)$, is an equivalence.

(b) Construct for any pointed type (X, x_0) an equivalence

$$\Big(\sum_{(P:X\to\mathcal{U})} P(x_0)\Big) \simeq \sum_{((A,a_0):\mathcal{U}_*)} (A, a_0) \to_* (X, x_0).$$

12.6 Show that any subuniverse is closed under equivalences, i.e., show that there is a map

$$(X \simeq Y) \to (P(X) \to P(Y))$$

for any subuniverse $P : \mathcal{U} \to \mathsf{Prop}$, and any $X, Y : \mathcal{U}$.

12.7 Show that the universe inclusions

$$\mathcal{U} \to \mathcal{U}^+ \qquad \text{and} \qquad \mathcal{U} \to \mathcal{U} \sqcup \mathcal{V}$$

defined in Remark 6.2.1, are embeddings.

13 Propositional truncations

The propositional truncation operation is a universal way of turning type a type A into a proposition $\|A\|$. Informally, the proposition $\|A\|$ is the proposition that A is inhabited. More precisely, the propositional truncation of A comes equipped with a map $A \to \|A\|$ and it is characterized by its universal property, which asserts that any map $A \to P$ into a proposition P extends uniquely to a map $\|A\| \to P$, as indicated in the diagram

$$\begin{tikzcd} A \arrow[d] \arrow[dr] & \\ \|A\| \arrow[r, dotted] & P. \end{tikzcd}$$

Using the propositional truncation operation we can define many important mathematical concepts, including the image of a map, surjectivity, and connected components. We will discuss those topics in §14.

13.1 The universal property of propositional truncations

Definition 13.1.1. Let A be a type, and let $f : A \to P$ be a map into a proposition P. We say that f is a **propositional truncation of** A if for every proposition Q, the precomposition map

$$- \circ f : (P \to Q) \to (A \to Q)$$

is an equivalence. This property of f is also called the **universal property of propositional truncation of** A

In other words, a map $f : A \to P$ into a proposition P is a propositional truncation of A if every map $g : A \to Q$ into a proposition extends uniquely along f, as indicated in the diagram

$$\begin{tikzcd} A \arrow[d, "f"'] \arrow[dr, "g"] & \\ P \arrow[r, dotted] & Q. \end{tikzcd}$$

Indeed, this unique extension property asserts that the type

$$\sum_{(h:P\to Q)} h \circ f = g$$

is contractible for every $g : A \to Q$. In other words, the unique extension property asserts that the precomposition function $- \circ f : (P \to Q) \to (A \to Q)$ is a contractible map, which is the case if and only if it is an equivalence.

Remark 13.1.2. Note that if Q is a proposition, then the type $X \to Q$ is a proposition for any type X. Furthermore, recall from Exercise 11.5.a that the map $(P \to Q) \to (A \to Q)$ is an equivalence as soon as there is a map in the converse direction. Therefore, in order to prove the universal property of the propositional truncation it suffices to show that

$$(A \to Q) \to (P \to Q).$$

We also note that the universal property of the propositional truncation of a type is formulated with respect to all propositions, regardless of the universe they live in.

Example 13.1.3. Suppose A is a type that comes equipped with a point $a : A$, such as the booleans, the type of natural numbers, or the loop space $\Omega(A)$ of a pointed type. Then the constant map

$$\mathsf{const}_\star : A \to \mathbf{1}$$

is a propositional truncation of A. To see this, let Q be an arbitrary proposition. It suffices to show that

$$(A \to Q) \to (\mathbf{1} \to Q).$$

To see this, let $f : A \to Q$. Then we have $f(a) : Q$, so we define $\mathsf{const}_{f(a)} : \mathbf{1} \to Q$. Thus we see that we have

$$\lambda f. \mathsf{const}_{f(a)} : (A \to Q) \to (\mathbf{1} \to Q).$$

This proves that $\mathsf{const}_\star : A \to \mathbf{1}$ satisfies the universal property of the propositional truncation of A.

Example 13.1.4. If the type A is already a proposition, then the identity function

$$\mathsf{id} : A \to A$$

is a propositional truncation of A. To see this, simply note that the precomposittion function with the identity function

$$- \circ \mathsf{id} : (A \to Q) \to (A \to Q)$$

is itself just the identity function. In particular, it is an equivalence.

Similarly, any equivalence $e : P \simeq P'$ between propositions satisfies the universal property of the propositional truncation of P, since precomposing by an equivalence is an equivalence by Theorem 11.4.1.

The universal property of the propositional truncation determines the propositional truncation up to equivalence. Such proofs of uniqueness from a universal property always follow the same pattern.

Proposition 13.1.5. *Let A be a type, and consider a commuting triangle*

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ P \arrow[rr, "h"'] & & P' \end{tikzcd}$$

where P and P' are propositions. If any two of the following three assertions hold, so does the third:

(i) The map f satisfies the universal property of the propositional truncation of A.

(ii) The map f' satisfies the universal propertyof the propositional truncation of A.

(iii) The map h is an equivalence.

Proof. Note that the map $h : P \to P'$ is an equivalence if and only if for every proposition Q, the precomposition map

$$- \circ h : (P' \to Q) \to (P \to Q)$$

is an equivalence. Thus, the claim follows by observing that for every proposition Q we have the triangle

$$\begin{tikzcd} (P' \to Q) \arrow[rr, "-\circ h"] \arrow[dr, "-\circ f'"'] & & (P \to Q) \arrow[dl, "-\circ f"] \\ & (A \to Q). & \end{tikzcd}$$

□

Corollary 13.1.6. *Consider two maps $f : A \to P$ and $f' : A \to P'$ into propositions P and P', both satisfying the universal property of the propositional truncation of A. Then the type of equivalences $e : P \simeq P'$ for which the triangle*

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ P \arrow[rr, "e"'] & & P' \end{tikzcd}$$

commutes, is contractible.

Remark 13.1.7. Note that the triangles in Proposition 13.1.5 and Corollary 13.1.6 always commutes, since P and P' are assumed to be propositions.

Now that we have shown that propositional truncations are determined uniquely, we will assume that any universe is closed under propositional truncations.

Axiom 13.1.8. *Any universe $\mathcal{U}$ is closed under propositional truncations: for any type $A : \mathcal{U}$ there is a proposition $\|A\| : \mathcal{U}$ equipped with a map $\eta : A \to \|A\|$ that satisfies the universal property of the propositional truncation.*

The propositional truncation is therefore an operation

$$\|-\| : \mathcal{U} \to \mathcal{U}$$

on the universe. One simple application of the universal property of the propositional truncation is that $\|-\|$ also acts on functions in a functorial way.

Proposition 13.1.9. *There is a map*

$$\|-\| : (A \to B) \to (\|A\| \to \|B\|)$$

for any two types A and B, such that

$$\|\mathsf{id}\| \sim \mathsf{id}$$

$$\|g \circ f\| \sim \|g\| \circ \|f\|.$$

Proof. For any $f : A \to B$, the map $\|f\| : \|A\| \to \|B\|$ is defined to be the unique extension

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\eta"'] & B \arrow[d, "\eta"] \\ \|A\| \arrow[r, dotted, "\|f\|"'] & \|B\|. \end{tikzcd}$$

To see that $\|-\|$ preserves identity maps and compositions, simply note that $\mathsf{id}_{\|A\|}$ is an extension of id_A, and that $\|g\| \circ \|f\|$ is an extension of $g \circ f$. Hence the homotopies are obtained by uniqueness. □

13.2 Propositional truncations as higher inductive types

The idea of higher inductive types is that types can be generated inductively not only by point constructors, such as $0_{\mathbb{N}} : \mathbb{N}$ and $\mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$, but also by path constructors. One of the first examples of a higher inductive type was the propositional truncation of a type A. This is a type $\|A\|$ equipped with one point constructor

$$\eta : A \to \|A\|,$$

one path constructor

$$\alpha : \prod_{(x,y:\|A\|)} x = y.$$

Note that the path constructor α immediately proves that $\|A\|$ is a proposition. Now we should formulate the induction principle for the propositional truncation.

Just as we did with the universal property, we will formulate the induction principle of the propositional truncation for an arbitrary map $f : A \to P$ into a proposition P. When the induction principle is formulated in this way, we will be able to show that f satisfies the universal property if and only if it satisfes the induction principle.

Consider a map $f : A \to P$ into a type equipped with a family of paths

$$\alpha : \prod_{(p,q:P)} p = q$$

witnessing that P is a proposition, and consider a type family B over P. The induction principle of the propositional truncation of A tells us what we have to do in order to construct a dependent function $h : \prod_{(p:P)} B(p)$.

In order to figure out what the induction principle has to be, we first note that if we start with a dependent function $h : \prod_{(p:P)} B(p)$, then we also obtain the function $h \circ f : \prod_{(x:A)} B(f(x))$. In other words, if we think of $f : A \to P$ as the point constructor of P, then the function $h \circ f$ describes the action of h on the points of P. The first requirement in the induction principle is therefore that B must come equipped with a dependent function

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

Furthermore, the function h acts on the paths in P via its dependent action on paths, which we constructed in Definition 5.4.2. The paths in P are generated by α, so we obtain a function

$$\lambda p.\, \lambda q.\, \mathsf{apd}_h(\alpha(p,q)) : \prod_{(p,q:P)} \mathsf{tr}_B(\alpha(p,q), h(p)) = h(q).$$

The induction principle must ensure that any function h defined via the induction principle, satisfies this law. Therefore, the second condition in the induction principle is that we must have a family of identifications

$$\prod_{(p,q:P)} \prod_{(y:B(p))} \prod_{(z:B(q))} \mathsf{tr}_B(\alpha(p,q), y) = z.$$

We now formulate the induction principle for propositional truncation.

Definition 13.2.1. Consider a map $f : A \to P$ into a type P equipped with a family of paths

$$\alpha : \prod_{(p,q:P)} p = q,$$

witnessing that P is a proposition. We say that f satisfies the induction principle for the propositional truncation of A if for any family B over P that comes equipped with

$$g : \prod_{(x:A)} B(f(x))$$

$$\beta : \prod_{(p,q:P)} \prod_{(y:B(p))} \prod_{(z:B(q))} \mathsf{tr}_B(\alpha(p,q), y) = z,$$

there is a dependent function $h : \prod_{(p:P)} B(p)$ equipped with a homotopy

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

In the following lemma we show that if a family B over $\|A\|$ comes equipped with a family of paths

$$\prod_{(x,y:\|A\|)} \prod_{(u:B(x))} \prod_{(v:B(y))} \mathsf{tr}_B(\alpha(x,y), u) = v,$$

then B must be a family of propositions.

Lemma 13.2.2. *Let P be a type equipped with a family of paths*

$$\alpha : \prod_{(p,q:P)} p = q,$$

showing that P is a proposition, and consider a type family B over P. The following are equivalent:

(i) The family B comes equipped with a family of identifications

$$\beta : \prod_{(p,q:P)} \prod_{(x:B(p))} \prod_{(y:B(q))} \mathsf{tr}_B(\alpha(p,q), x) = y,$$

(ii) The family B is a family of propositions.

Proof. Assuming that (i) holds, we will show that each $B(p)$ is a proposition by showing that

$$B(p) \to \mathsf{is\text{-}contr}(B(p)).$$

Let $x : B(p)$. We have to construct a center of contraction and a contraction. Our plan is to use β to define the contraction, so it is natural to define the center as $\mathsf{tr}_B(\alpha(p,p), x)$. Now we take

$$\beta(p,p,x) : \prod_{(y:B(p))} \mathsf{tr}_B(\alpha(p,p), x) = y$$

as our contraction. This completes the proof that B is a family of propositions.

The converse is immediate: if B is a family of propositions, then any two terms in any $B(q)$ can be identified. $\square$

Definition 13.2.3. Consider a map $f : A \to P$ into a proposition P. We say that f satisfies the dependent universal property of the propositional truncation of A, if for any family Q of propositions over P, the precomposition map

$$- \circ f : \Big(\prod_{(p:P)} Q(p)\Big) \to \Big(\prod_{(x:A)} Q(f(x))\Big)$$

is an equivalence.

Theorem 13.2.4. *Consider a map $f : A \to P$ into a proposition P. Then the following are equivalent:*

(i) The map f is a propositional truncation.

(ii) The map f satisfies the dependent universal property of the propositional truncation.

(iii) The map f satisfies the induction principle of the propositional truncation.

Proof. We will first show that (i) and (ii) are equivalent. Of course, the universal property is a special case of the dependent universal property, so the fact that (ii) implies (i) is immediate. We now show that (i) implies (ii). Let Q be a family of propositions over P, and consider the following commuting diagram:

$$\begin{tikzcd}
\left(\sum_{(h:P\to P)}\prod_{(p:P)}Q(h(p))\right) \arrow[r, "\mathsf{tot}_{-\circ f}(-\circ f)"] \arrow[d, "\mathsf{choice}^{-1}"'] & \left(\sum_{(g:A\to P)}\prod_{(x:A)}Q(g(x))\right) \arrow[d, "\mathsf{choice}^{-1}"] \\
\left(P \to \sum_{(p:P)} Q(p)\right) \arrow[r, "-\circ f"] \arrow[d, "\mathsf{pr}_1\circ -"'] & \left(A \to \sum_{(p:P)} Q(p)\right) \arrow[d, "\mathsf{pr}_1\circ -"] \\
\left(P \to P\right) \arrow[r, "-\circ f"'] & \left(A \to P\right)
\end{tikzcd}$$

In this diagram the bottom map is an equivalence by the universal property of the propositional truncation of A. Note also that the type $\sum_{(p:P)} Q(p)$ is a proposition by Exercise 10.3, so it follows that also the middle map is an equivalence. Furthermore, the type theoretic choice maps are equivalences by Theorem 11.2.1, so it also follows that the top map is an equivalence. Now we use Theorem 9.1.6 to conclude that the family of maps

$$-\circ f : \left(\prod_{(p:P)}Q(h(p))\right) \to \left(\prod_{(x:A)}Q(h(f(x)))\right)$$

indexed by $h : P \to P$ is a family of equivalences. The dependent universal property is now just a special case: take $h \equiv \mathsf{id}$. This completes the proof that (i) is equivalent to (ii).

It remains to show that (ii) is equivalent to (iii). By Lemma 13.2.2 it follows that the induction principle is equivalent to the property that for each family Q of propositions over P, the precomposition map

$$-\circ f : \left(\prod_{(p:P)}Q(p)\right) \to \left(\prod_{(x:A)}Q(f(x))\right)$$

has a section. Since the domain and codomain of this map are propositions by Theorem 11.1.3, we see that this precomposition map has a section if and only if it is an equivalence. □

Exercises

13.1 Let A be a type and let P be a proposition, and suppose that P is a retract of A. Show that the retraction $A \to P$ is a propositional truncation.

13.2 Consider two maps $f : A \to P$ and $g : B \to Q$ into propositions P and Q. Recall from Exercise 10.3 that the type $P \times Q$ is also a proposition. Show that if both f and g are propositional truncations then the map $f \times g : A \times B \to P \times Q$ is also a propositional truncation. Conclude that

$$\|A \times B\| \simeq \|A\| \times \|B\|.$$

13.3 Consider two propositions P and Q, and define

$$\begin{aligned} P \wedge Q &:\equiv P \times Q \\ P \vee Q &:\equiv \|P + Q\|. \end{aligned}$$

(a) Construct maps $i : P \to P \vee Q$ and $j : Q \to P \vee Q$.

(b) Prove the universal property of disjunction, i.e., show that for any proposition R, the map

$$(P \vee Q \to R) \to (P \to R) \wedge (Q \to R)$$

given by $h \mapsto (h \circ i, h \circ j)$ is an equivalence.

13.4 Consider a family P of propositions over a type A, and define

$$\begin{aligned}\forall_{(x:A)} P(x) &:\equiv \prod_{(x:A)} P(x) \\ \exists_{(x:A)} P(x) &:\equiv \left\| \sum_{(x:A)} P(x) \right\|\end{aligned}$$

(a) Construct a map $i_a : P(a) \to \exists_{(x:A)} P(x)$ for each $a : A$.
(b) Prove the universal property of the existential quantification, i.e. show that for any proposition Q, the map

$$\Big(\Big(\exists_{(x:A)} P(x)\Big) \to Q\Big) \to \Big(\forall_{(x:A)} (P(x) \to Q)\Big)$$

given by $h \mapsto \lambda x.\, h \circ i_x$, is an equivalence.

13.5 Show that

$$\|A\| \simeq \prod_{(P:\mathsf{Prop})} (A \to P) \to P$$

for any type $A : \mathcal{U}$. This is called the **impredicative encoding** of the propositional truncation.

14 The image of a map and the replacement axiom

The idea of the image of a map $f : A \to X$ is that it is, in a way, the least subtype of X that contains all the values of f. More precisely, the image of f is an embedding $i : \mathrm{im}(f) \hookrightarrow X$ that fits in a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & \mathrm{im}(f) \arrow[dl, hook, "i"] \\ & X & \end{tikzcd}$$

and satisfies the *universal property* of the image of f. The universal property of the image of f asserts that if a subtype $B \hookrightarrow X$ contains all the values of f, then it contains the image of f. The image of a map can be constructed using the propositional truncation operation. In fact, we can also go the other way around: The propositional truncation of a type A is the image of the map $A \to \mathbf{1}$.

The final topic of this section is the type theoretic replacement axiom. A specific instance of the replacement axiom asserts that the image of any map $f : A \to \mathcal{U}$ is equivalent to a type in $\mathcal{U}$, provided that A is equivant to a type in $\mathcal{U}$. This property will be used to construct quotients in type theory, much in the same way as quotients are constructed in set theory.

We should note that the existence of the propositional truncation operation and the replacement axiom will be assumed for now. However, once we assume that universes are closed under pushouts, we will be able to construct the propositional truncations and we will be able to prove the replacement axiom. These constructions will be given in **??**.

14.1 The image of a map

Definition 14.1.1. Let $f : A \to X$ and $g : B \to X$ be maps. A **morphism** from f to g over X consists of a map $h : A \to B$ equipped with a homotopy $H : f \sim g \circ h$ witnessing that the triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

commutes. Thus, we define the type

$$\mathrm{hom}_X(f,g) :\equiv \textstyle\sum_{(h:A\to B)} f \sim g \circ h.$$

Composition of morphisms over X is defined by

$$(k,K) \circ (h,H) :\equiv (k \circ h, H \cdot (K \cdot h)).$$

Definition 14.1.2. Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & I \arrow[dl, "i"] \\ & X & \end{tikzcd}$$

with $H : f \sim i \circ q$, where i is an embedding. We say that i has the **universal property of the image of** f if the map

$$- \circ (q,H) : \mathrm{hom}_X(i,m) \to \mathrm{hom}_X(f,m)$$

is an equivalence for every embedding $m : B \to X$.

Remark 14.1.3. Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & I \arrow[dl, "i"] \\ & X & \end{tikzcd}$$

with $H : f \sim i \circ q$, where i is an embedding. Then it is not hard to see that the embedding i satisfies the universal property of the image inclusion if and only if for every commuting triangle

$$\begin{tikzcd} A \arrow[rr, "g"] \arrow[dr, "f"'] & & B \arrow[dl, "m"] \\ & X & \end{tikzcd}$$

with $G : f \sim m \circ g$, where m is an embedding, the type of quadruples (h, K, L, M) consisting of

(i) a map $h : I \to B$,

(ii) a homotopy $K : i \sim m \circ h$ witnessing that the triangle

$$\begin{tikzcd} I \arrow[rr, "h"] \arrow[dr, "i"'] & & B \arrow[dl, "m"] \\ & X & \end{tikzcd}$$

commutes,

(iii) a homotopy $L : g \sim h \circ q$ witnessing that the triangle

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "g"'] & & I \arrow[dl, "h"] \\ & B & \end{tikzcd}$$

commutes,

(iv) a homotopy $M : H \cdot (K \cdot q) \sim G \cdot (m \cdot L)$ witnessing that the square

$$\begin{tikzcd} f \arrow[r, "G"] \arrow[d, "H"'] & m \circ g \arrow[d, "m \cdot L"] \\ i \circ q \arrow[r, "K \cdot q"'] & m \circ h \circ g \end{tikzcd}$$

commutes,

is contractible. However, the situation is in fact much simpler, because the type $\hom_X(f, m)$ is a proposition whenever m is an embedding.

Remark 14.1.4. Suppose that the map $f : A \to X$ has a section. Then the identity function

$$\mathsf{id} : X \to X$$

satisfies the universal property of the image of f.

Remark 14.1.5. Suppose that $f : A \to X$ is already an embedding. Then f itself satisfies the universal property of the image of f.

Lemma 14.1.6. *For any $f : A \to X$ and any embedding $m : B \to X$, the type $\hom_X(f, m)$ is a proposition.*

Proof. Recall from Exercise 11.15 that the type $\hom_X(f, m)$ is equivalent to the type

$$\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_m(x).$$

Therefore it suffices to show that this type is a proposition. Recall from Corollary 10.3.7 that a map is an embedding if and only if its fibers are propositions. Thus we see that the type $\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_m(x)$ is a product of propositions, hence it is a proposition by Theorem 11.1.3. □

Proposition 14.1.7. *Consider a commuting triangle*

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & I \arrow[dl, "i"] \\ & X & \end{tikzcd}$$

with $H : f \sim i \circ q$, where i is an embedding. Then the following are equivalent:

(i) The embedding i satisfies the universal property of the image inclusion of f.

(ii) For every embedding $m : B \to X$ there is a map

$$\hom_X(f, m) \to \hom_X(i, m).$$

Proof. Since $\hom_X(f, m)$ is a proposition for every every embedding $m : B \to X$, the claim follows immediately by Exercise 11.5. □

Just as in the cases for pullbacks and pushouts, the universal property of the image implies that the image is determined uniquely. We will show here that the type of image factorizations of any map is a proposition. In §14.1 we will construct the image, after constructing the propositional truncation.

Proposition 14.1.8. *Let f be a map, and consider two commuting triangles*

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & B \arrow[dl, "i"] \\ & X & \end{tikzcd} \qquad \begin{tikzcd} A \arrow[rr, "q'"] \arrow[dr, "f"'] & & B' \arrow[dl, "i'"] \\ & X & \end{tikzcd}$$

with $I : f \sim i \circ q$ and $I' : f \sim i' \circ q'$, in which i and i' are assumed to be embeddings. Moreover, consider

$$(h, H) : \hom_X(i, i')$$

equipped with an identification $(h, H) \circ (q, I) = (q', I')$ in $\hom_X(f, i')$. Then, if any two of the following properties hold, so does the third:

(i) The embedding i satisfies the universal property of the image inclusion of f.

(ii) The embedding i' satisfies the universal property of the image inclusion of f.

(iii) The map h is an equivalence.

Proof. Consider an embedding $m : C \to X$. Then we have a commuting triangle

$$\begin{tikzcd} \hom_X(i', m) \arrow[rr, "- \circ (h,H)"] \arrow[dr, "- \circ (q', I')"'] & & \hom_X(i, m) \arrow[dl, "- \circ (q, I)"] \\ & \hom_X(f, m), & \end{tikzcd}$$

so it follows that if any two of these maps are equivalences, then so is the third. The claim now follows by the observation that $- \circ (h, H)$ is an equivalence for every embedding $m : C \to X$ if and only if h is an equivalence. □

Corollary 14.1.9. *Consider two image factorizations*

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & B \arrow[dl, "i"] \\ & X & \end{tikzcd} \qquad \begin{tikzcd} A \arrow[rr, "q'"] \arrow[dr, "f"'] & & B' \arrow[dl, "i'"] \\ & X & \end{tikzcd}$$

of a map f, with $I : f \sim i \circ q$ and $I' : f \sim i' \circ q'$. Then the type of $(e, H) : \hom_X(i, i')$ in which e is an equivalence, equipped with an identification

$$(e, H) \circ (q, I) = (q', I')$$

in $\hom_X(f, i')$, is contractible.

The image of a map $f : A \to X$ can now be defined using the propositional truncation:

Definition 14.1.10. For any map $f : A \to X$ we define the **image** of f to be the type

$$\mathrm{im}(f) :\equiv \sum_{(x:X)} \|\mathsf{fib}_f(x)\|.$$

Furthermore, we define:

(i) The **image inclusion**

$$i_f : \mathrm{im}(f) \to X$$

to be the projection pr_1.

(ii) The map

$$q_f : A \to \mathrm{im}(f)$$

to be the map given by $q_f(x) :\equiv (f(x), \eta(x, \mathsf{refl}_{f(x)}))$.

(iii) The homotopy $I_f : f \sim i_f \circ q_f$ witnessing that the triangle

$$\begin{tikzcd} A \arrow[rr, "q_f"] \arrow[dr, "f"'] & & \mathrm{im}(f) \arrow[dl, "i_f"] \\ & X & \end{tikzcd}$$

commutes, to be given by $I_f(x) :\equiv \mathsf{refl}_{f(x)}$.

Proposition 14.1.11. *The image inclusion $i_f : \mathrm{im}(f) \to X$ of any map $f : A \to X$ is an embedding.*

Proof. The fiber of i_f at $x : X$ is equivalent to the type $\|\mathsf{fib}_f(x)\|$. In particular we see that the fibers are propositions, so i_f is an embedding. □

Theorem 14.1.12. *The image inclusion $i_f : \mathrm{im}(f) \to X$ of any map $f : A \to X$ satisfies the universal property of the image inclusion of f.*

Proof. Consider an embedding $m : B \to X$. Note that we have a commuting square

$$\begin{tikzcd} \hom_X(i_f, m) \arrow[r] \arrow[d] & \hom_X(f, m) \arrow[d] \\ \Big(\prod_{(x:X)} \mathsf{fib}_{i_f}(x) \to \mathsf{fib}_m(x)\Big) \arrow[r, "h \mapsto \lambda x.\, h_x \circ \varphi_x"'] & \Big(\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_m(x)\Big) \end{tikzcd}$$

The vertical maps are of the form

$$(h, H) \mapsto \lambda x.\, \lambda(y, p).\, (h(y), H(y)^{-1} \cdot p),$$

and they are both equivalences. The map

$$\varphi_x : \mathsf{fib}_f(x) \to \mathsf{fib}_{i_f}(x)$$

given by $\varphi_x(a, p) :\equiv ((h(a), \eta(a, p)), p)$ is a propositional truncation for every $x : X$. Therefore it follows that the map

$$(\mathsf{fib}_{i_f}(x) \to \mathsf{fib}_m(x)) \to (\mathsf{fib}_f(x) \to \mathsf{fib}_m(x))$$

is an equivalence, for every $x : X$. Thus we conclude that the bottom map in the above square is an equivalence, which implies that the top map is an equivalence. □

Example 14.1.13. An important special case of the homotopy image of a map is the image of the terminal projection

$$\mathsf{const}_\star : A \to \mathbf{1},$$

which results in an embedding $I \hookrightarrow \mathbf{1}$. Embeddings into the unit type are in fact just propositions. To see this, note that

$$\begin{aligned}
\textstyle\sum_{(A:\mathcal{U})}\sum_{(f:A\to\mathbf{1})}\mathsf{is\text{-}emb}(f) &\simeq \textstyle\sum_{(A:\mathcal{U})}\mathsf{is\text{-}emb}(\mathsf{const}_\star) \\
&\simeq \textstyle\sum_{(A:\mathcal{U})}\prod_{(x:\mathbf{1})}\mathsf{is\text{-}prop}(\mathsf{fib}_{\mathsf{const}_\star}(x)) \\
&\simeq \textstyle\sum_{(A:\mathcal{U})}\mathsf{is\text{-}prop}(\mathsf{fib}_{\mathsf{const}_\star}(\star)) \\
&\simeq \textstyle\sum_{(A:\mathcal{U})}\mathsf{is\text{-}prop}(A).
\end{aligned}$$

Therefore, the universal property of the image of the map $A \to \mathbf{1}$ is equivalently described as a proposition P satisfying the universal property of the propositional truncation.

14.2 Surjective maps

Another application of the propositional truncation is the notion of surjective map.

Definition 14.2.1. A map $f : A \to B$ is said to be **surjective** if there is a term of type

$$\mathsf{is\text{-}surj}(f) :\equiv \textstyle\prod_{(y:B)}\|\mathsf{fib}_f(b)\|.$$

Example 14.2.2. Any equivalence is a surjective map, and so is any map that has a section (those are sometimes called **split epimorphisms**). Other examples include the base point inclusion $\mathbf{1} \to \mathbf{S}^n$ for any $n \geq 1$.

Proposition 14.2.3. *Consider a map $f : A \to B$. Then the following are equivalent:*

(i) The map $f : A \to B$ is surjective.

(ii) For any family P of propositions over B, the precomposition map

$$-\circ f : \left(\textstyle\prod_{(y:B)}P(y)\right) \to \left(\textstyle\prod_{(x:A)}P(f(x))\right)$$

is an equivalence.

Proof. Suppose first that f is surjective, and consider the commuting square

$$\begin{tikzcd}
\left(\prod_{(y:B)}P(y)\right) \arrow[r, "-\circ f"] \arrow[d, "h\mapsto\lambda y.\,\mathsf{const}_{h(y)}"'] & \left(\prod_{(x:A)}P(f(x))\right) \\
\left(\prod_{(y:B)}\|\mathsf{fib}_f(y)\| \to P(y)\right) \arrow[r, "h\mapsto\lambda y.\,h(y)\circ\eta"'] & \left(\prod_{(y:B)}\mathsf{fib}_f(y) \to P(y)\right) \arrow[u, "h\mapsto\lambda x.\,h(f(x),(x,\mathsf{refl}_{f(x)}))"']
\end{tikzcd}$$

In this square, the bottom map is an equivalence by the universal property of the propositional truncation of $\mathsf{fib}_f(y)$. The map on the right is also easily seen to be an equivalence. Furthermore, the map on the left is an equivalence by the assumption that f is surjective, from which it follows that the types $\|\mathsf{fib}_f(y)\|$ are contractible. Therefore it follows that the top map is an equivalence, which completes the proof that (i) implies (ii).

For the converse, it follows immediately from the assumption (ii) that

$$-\circ f : \Big(\prod_{(y:B)} \|\mathsf{fib}_f(y)\|\Big) \to \Big(\prod_{(x:A)} \|\mathsf{fib}_f(f(x))\|\Big)$$

is an equivalence. Hence it suffices to construct a term of type $\|\mathsf{fib}_f(f(x))\|$ for each $x : A$. This is easy, because we have

$$\eta(x, \mathsf{refl}_{f(x)}) : \|\mathsf{fib}_f(f(x))\|. \qquad \square.$$

Theorem 14.2.4. *Consider a commuting triangle*

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "f"'] & & B \arrow[dl, "m"] \\ & X & \end{tikzcd}$$

in which m is an embedding. Then the following are equivalent:

(i) The embedding m satisfies the universal property of the image inclusion of f.

(ii) The map q is surjective.

Proof. First assume that m satisfies the universal property of the image inclusion of f, and consider the composite function

$$\Big(\sum_{(y:B)} \|\mathsf{fib}_q(y)\|\Big) \xrightarrow{\mathsf{pr}_1} B \xrightarrow{m} X.$$

Note that $m \circ \mathsf{pr}_1$ is a composition of embeddings, so it is an embedding. By the universal property of m there is a unique map h for which the triangle

$$\begin{tikzcd} B \arrow[rr, dotted, "h"] \arrow[dr, "m"'] & & \sum_{(y:B)} \|\mathsf{fib}_q(y)\| \arrow[dl, "m\circ\mathsf{pr}_1"] \\ & X & \end{tikzcd}$$

commutes. Now note that $\mathsf{pr}_1 \circ h$ is a map such that $m \circ (\mathsf{pr}_1 \circ h) \sim m$. The identity function is another map for which we have $m \circ \mathsf{id} \sim m$, so it follows by uniqueness that $\mathsf{pr}_1 \circ h \sim \mathsf{id}$. In other words, the map h is a section of the projection map. Therefore we obtain by Exercise 11.18 a dependent function

$$\prod_{(b:B)} \|\mathsf{fib}_q(b)\|,$$

showing that q is surjective.

For the converse, suppose that q is surjective. To prove that m satisfies the universal property of the image factorization of f, it suffices to construct an equivalence

$$\hom_X(f, m') \to \hom_X(m, m'),$$

for any embedding $m' : B' \to X$. To see that there is such an equivalence, we make the following calculation

$$\hom_X(m, m') \simeq \prod_{(x:X)} \mathsf{fib}_m(x) \to \mathsf{fib}_{m'}(x)$$

$$\begin{aligned}
&\simeq \prod_{(b:B)} \mathsf{fib}_{m'}(m(b)) \\
&\simeq \prod_{(a:A)} \mathsf{fib}_{m'}(m(q(a))) \\
&\simeq \prod_{(a:A)} \mathsf{fib}_{m'}(f(a)) \\
&\simeq \prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_{m'}(x) \\
&\simeq \hom_X(f, m').
\end{aligned}$$

In this calculation, the first and last equivalence hold by Exercise 11.15. The second and second to last equivalences hold by Exercise 11.14. The third equivalence holds by Proposition 14.2.3, since q is assumed to be surjective, and the fourth equivalence holds since we have a homotopy $f \sim m \circ f$. □

Corollary 14.2.5. *Every map factors uniquely as a surjective map followed by an embedding.*

Proof. Consider a map $f : A \to X$, and two factorizations

$$\begin{tikzcd} A \arrow[r, "q"] \arrow[dr, "f"'] & B \arrow[d, "i"] \\ & X \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "q'"] \arrow[dr, "f"'] & B' \arrow[d, "i'"] \\ & X \end{tikzcd}$$

of f where m and m' are embeddings, and q and q' are surjective. Then both m and m' satisfy the universal property of the image factorization of f by Theorem 14.2.4. Now it follows by Corollary 14.1.9 that the type of $(e, H) : \hom_X(i, i')$ in which e is an equivalence, equipped with an identification

$$(e, H) \circ (q, I) = (q', I')$$

in $\hom_X(f, i')$, is contractible. □

14.3 Type theoretic replacement

Definition 14.3.1. (i) A type A is said to be **essentially small** if there is a type $X : \mathcal{U}$ and an equivalence $A \simeq X$. We write

$$\mathsf{ess\text{-}small}(A) :\equiv \textstyle\sum_{(X:\mathcal{U})} A \simeq X.$$

(ii) A map $f : A \to B$ is said to be **essentially small** if for each $b : B$ the fiber $\mathsf{fib}_f(b)$ is essentially small. We write

$$\mathsf{ess\text{-}small}(f) :\equiv \textstyle\prod_{(b:B)} \mathsf{ess\text{-}small}(\mathsf{fib}_f(b)).$$

(iii) A type A is said to be **locally small** if for every $x, y : A$ the identity type $x = y$ is essentially small. We write

$$\mathsf{loc\text{-}small}(A) :\equiv \textstyle\prod_{(x,y:A)} \mathsf{ess\text{-}small}(x = y).$$

Example 14.3.2. (i) Any essentially $\mathcal{U}$-small type is also locally $\mathcal{U}$-small.

(ii) Any univalent universe $\mathcal{U}$ is locally $\mathcal{U}$-small, because by the univalence axiom we have equivalences

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

for each $A, B : \mathcal{U}$, and the type $A \simeq B$ is in $\mathcal{U}$.

(iii) Any proposition is locally small with respect to any universe $\mathcal{U}$.

(iv) For any family P of locally $\mathcal{U}$-small types over a essentially $\mathcal{U}$-small type A, the dependent product $\prod_{(x:A)} P(x)$ is locally $\mathcal{U}$-small. In particular, any type $A \to B$ of functions from an essentially small type into a locally small type is again locally small.

Lemma 14.3.3. *The type* $\mathsf{ess\text{-}small}(A)$ *is a proposition for any type* A*.*

Proof. Let A be a type, not necessarily in $\mathcal{U}$. In order to show that $\mathsf{ess\text{-}small}(A)$ is a proposition, we will use Theorem 10.1.3 and show that for any $X : \mathcal{U}$ and any equivalence $e : A \simeq X$, the type

$$\textstyle\sum_{(Y:\mathcal{U})} A \simeq Y$$

is contractible. Note that we have an equivalence

$$\left(\textstyle\sum_{(Y:\mathcal{U})} X \simeq Y\right) \simeq \left(\textstyle\sum_{(Y:\mathcal{U})} A \simeq Y\right)$$

because precomposing with the equivalence $e : A \simeq X$ is an equivalence. However, the type $\sum_{(Y:\mathcal{U})} X \simeq Y$ is contractible by Theorem 12.1.2. This shows that $\mathsf{ess\text{-}small}(A)$ is equivalent to a contractible type, assuming that A is essentially small. □

Corollary 14.3.4. *For each function* $f : A \to B$*, the type* $\mathsf{ess\text{-}small}(f)$ *is a proposition, and for each type* X *the type* $\mathsf{loc\text{-}small}(X)$ *is a proposition.*

Proof. This follows from the fact that propositions are closed under dependent products, established in Theorem 11.1.3. □

Recall that in set theory, the replacement axiom asserts that for any family of sets $\{X_i\}_{i \in I}$ indexed by a set I, there is a set $X[I]$ consisting of precisely those sets x for which there exists an $i \in I$ such that $x \in X_i$. In other words: the image of a set-indexed family of sets is again a set. Without the replacement axiom, $X[I]$ would be a class. In the following corollary we establish a type-theoretic analogue of the replacement axiom: the image of a family of small types indexed by a small type is again (essentially) small.

Axiom 14.3.5. *For any map* $f : A \to B$ *from an essentially small type* A *into a locally small type* B*, the image of* f *is again essentially small.*

Example 14.3.6. For any type $A : \mathcal{U}$, the image of the constant map $\mathsf{const}_A : \mathbf{1} \to \mathcal{U}$ is essentially small. This image is called the **connected component** of the universe at A. To see why, let us calculate

$$\begin{aligned} \mathrm{im}(\mathsf{const}_A) &\equiv \textstyle\sum_{(X:\mathcal{U})} \|\mathsf{fib}_{\mathsf{const}_A}(X)\| \\ &\equiv \textstyle\sum_{(X:\mathcal{U})} \left\|\textstyle\sum_{(t:\mathbf{1})} A = X\right\| \\ &\simeq \textstyle\sum_{(X:\mathcal{U})} \|A = X\|. \end{aligned}$$

We see that the image of $\mathsf{const}_A : \mathbf{1} \to \mathcal{U}$ is the type of all types that are *merely* equal to A. In other words, they are equal to A in an unspecified way.

Example 14.3.7. The type $\mathbb{F}$ of all finite types is defined to be the image of the map

$$\mathsf{Fin} : \mathbb{N} \to \mathcal{U}_0$$

By the replacement axiom, this type is essentially small.

Exercises

14.1 Consider a map $f : A \to P$ into a proposition P. Show that the following are equivalent:
   (i) The map f is a propositional truncation of A.
   (ii) The map f is surjective.

14.2 Consider a map $f : A \to B$. Show that the following are equivalent:
   (i) f is an equivalence.
   (ii) f is both surjective and an embedding.

14.3 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$, and assume that h is surjective. Show that the following are equivalent:
   (i) The map f is surjective.
   (ii) The map g is surjective.

14.4 Consider a map $f : A \to B$. Show that the following are equivalent:
   (i) The map f is surjective.
   (ii) For every set C, the precomposition function

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

   is an embedding.

Hint: To show that (ii) implies (i), use the assumption with the set $C \equiv \mathsf{Prop}_{\mathcal{U}}$, where $\mathcal{U}$ is a univalent universe containing both A and B.

14.5 Let us say that a type family B over A is **univalent** if the map

$$(x = y) \to (B(x) \simeq B(y))$$

is an equivalence, for every $x, y : A$.
   (a) Show that a family $B : A \to \mathcal{U}$ is univalent if and only if the map $B : A \to \mathcal{U}$ is an embedding.
   (b) For any family $B : A \to \mathcal{U}$, show that the type family $\hat{B} : \hat{A} \to \mathcal{U}$ defined by

$$\hat{A} :\equiv \mathrm{im}(B)$$
$$\hat{B}(X, p) :\equiv X$$

   is univalent.
   (c) For any two families $B : A \to \mathcal{U}$ and $D : C \to \mathcal{V}$, define the type of **cartesian morphisms**

$$\mathsf{cart\text{-}hom\text{-}Fam}((A, B), (C, D)) :\equiv \textstyle\sum_{(f : A \to C)} \prod_{(x:A)} B(x) \simeq D(f(x)).$$

   Construct a cartesian morphism

$$(\eta, \alpha) : \mathsf{cart\text{-}hom\text{-}Fam}((A, B), (\hat{A}, \hat{B})).$$

(d) Show that for any family $B : A \to \mathcal{U}$ and any *univalent* family $D : C \to \mathcal{V}$, the map

$$\mathsf{cart\text{-}hom\text{-}Fam}((\hat{A}, \hat{B}), (C, D)) \to \mathsf{cart\text{-}hom\text{-}Fam}((A, B), (C, D))$$

given by

$$(f, e) \mapsto (f \circ \eta, \lambda x.\, e_x \circ \alpha_x)$$

is an equivalence. This is the **universal property** of the univalent completion of A.

Chapter III

Topics in univalent mathematics

15 Elementary number theory

One of the things type theory is great for, is for the formalization of mathematics in a computer proof assistant. Those are programs that can compile any type theoretical construction to check that this construction indeed has the type it was claimed it has.

At this point in our development of type theory there are two areas of mathematics that would be natural to try to do in type theory: discrete mathematics and elementary number theory. Indeed, how does one define in type theory the greatest common divisor of two natural numbers, or how does one show that there are infinitely many primes? How does one even formalize that every non-empty subset of the natural numbers has a least element?

To answer these questions we will run into questions of decidability. How do we write a term that decides wheter a number is prime or not? Or indeed, is it even true that every non-empty subset of the natural numbers has a least element? What about the subset of $\mathbb{N}$ that contains 1, and it contains 0 if and only if Goldbach's conjecture holds? Finding the least element of this subset is equivalent to settling the conjecture!

Therefore, we will prove the well-foundedness of the natural numbers for decidable subsets of $\mathbb{N}$. In fact, we will show it for decidable families, because sometimes we don't know in advance whether a family of types is in fact a subtype. A consequence of involving decidability in the well-foundedness of the natural numbers is that for many properties one has to prove that they are decidable. Luckily this is the case: many of the familiar properties that one encounters in number theory are indeed decidable.

15.1 Decidability

A common way of reasoning in mathematics is via a proof by contradiction: "in order to show that P holds we show that it cannot be the case that P doesn't hold". There are no inference rules in type theory that allow us to obtain a term of type P from a term of type $\neg\neg P$. However, for some propositions P one can construct a function $\neg\neg P \to P$. The *decidable propositions* from a class of such propositions P for which we can show $\neg\neg P \to P$.

The following definition of decidability is made for general types, even though we will mostly be interested in the decidabilyt of proposition. The reason will become aparent in a moment, when we show that types with decidable equality are sets. This useful theorem would become trivial if we restricted the notion of decidability to propositions.

Definition 15.1.1. A type A is said to be decidable if it comes equipped with a term of type

$$\mathsf{is\text{-}decidable}(A) :\equiv A + \neg A.$$

Decidable propositions are called **classical**. We will write

$$\mathsf{classical\text{-}Prop}_{\mathcal{U}} :\equiv \sum_{(P:\mathsf{Prop}_{\mathcal{U}})} \mathsf{is\text{-}decidable}(P)$$

for the type of all classical propositions (with respect to a universe $\mathcal{U}$).

Example 15.1.2. The types $\mathbf{1}$ and $\emptyset$ are decidable. Indeed, we have

$$\begin{aligned} \mathsf{dec\text{-}}\mathbf{1} &:\equiv \mathsf{inl}(\star) && : \mathsf{is\text{-}decidable}(\mathbf{1}) \\ \mathsf{dec\text{-}}\emptyset &:\equiv \mathsf{inr}(\mathsf{id}) && : \mathsf{is\text{-}decidable}(\emptyset). \end{aligned}$$

Any type A equipped with a point $a : A$ is decidable.

Since P and $\neg P$ are mutually exclusive cases, it follows that $\mathsf{is\text{-}decidable}(P)$ is a proposition. Therefore we see that the type of decidable propositions in a universe $\mathcal{U}$ form a subtype of the type of all propositions in $\mathcal{U}$.

Lemma 15.1.3. *For any proposition P, the type $\mathsf{is\text{-}decidable}(P)$ is a proposition.*

Proof. By Theorem 10.1.3 it suffices to show that

$$\prod_{(t,t':\mathsf{is\text{-}decidable}(P))} t = t.$$

We proceed by case analysis on t and t'. We have four cases to consider:

$$\begin{aligned} \mathsf{inl}(p) &= \mathsf{inl}(p') && \mathsf{inr}(f) = \mathsf{inl}(p') \\ \mathsf{inl}(p) &= \mathsf{inr}(f') && \mathsf{inr}(f) = \mathsf{inr}(f'). \end{aligned}$$

We construct these four identifications as follows:

(i) First, we want to show that $\mathsf{inl}(p) = \mathsf{inl}(p')$ for any $p, p' : P$. We obtain this identification from the fact that $p = p'$, which we have because P is assumed to be a proposition.

(ii) Next, we want to show that $\mathsf{inl}(p) = \mathsf{inr}(f')$ for any $p : P$ and $f' : \neg P$. Since we have contradictory assumptions, we obtain $f'(p) : \emptyset$. We now obtain the desired identification by applying the function $\emptyset \to (\mathsf{inl}(p) = \mathsf{inr}(f')$.

(iii) The construction of an identification $\mathsf{inr}(f) = \mathsf{inl}(p')$ for $f : \neg P$ and $p' : P$ is similar. We have $f(p') : \emptyset$, which gives the desired identification via the function $\emptyset \to (\mathsf{inr}(f) = \mathsf{inl}(p'))$.

(iv) Finally, we want to show that $\mathsf{inr}(f) = \mathsf{inr}(f')$ for $f, f' : \neg P$. The type $\neg P$ is a proposition, so we have an identification $f = f'$ from which we obtain $\mathsf{inr}(f) = \mathsf{inr}(f')$. □

We have seen in Theorem 12.3.1 that the univalence axiom implies propositional extensionality. Recall that propositional extensionality is the property that the map

$$(P = Q) \to (P \leftrightarrow Q)$$

is an equivalence. We will use this fact here to conclude that $\mathsf{classical\text{-}Prop}_{\mathcal{U}}$ is equivalent to $\mathbf{2}$.

Proposition 15.1.4. *The type of classical propositions in any universe $\mathcal{U}$ is equivalent to* $\mathbf{2}$.

Proof. Since the empty type and the unit type are decidable propositions, we have a map $\varphi : \mathbf{2} \to \mathsf{classical\text{-}Prop}_{\mathcal{U}}$ defined by

$$\varphi(1_{\mathbf{2}}) :\equiv (\mathbf{1}, \mathsf{dec}\text{-}\mathbf{1})$$
$$\varphi(0_{\mathbf{2}}) :\equiv (\emptyset, \mathsf{dec}\text{-}\emptyset).$$

Next, we define a map $\psi : \mathsf{classical\text{-}Prop}_{\mathcal{U}} \to \mathbf{2}$. Let P be a proposition that comes equipped with a term $t : P + \neg P$. To define a boolean, we proceed by case analysis on t. The map ψ is thus defined by

$$\psi(P, \mathsf{inl}(p)) :\equiv 1_{\mathbf{2}}$$
$$\psi(P, \mathsf{inr}(f)) :\equiv 0_{\mathbf{2}}.$$

To see that ψ is an inverse of φ, note that

$$\varphi(\psi(P, \mathsf{inl}(p))) \equiv (\mathbf{1}, \mathsf{dec}\text{-}\mathbf{1}) \qquad \psi(\varphi(1_{\mathbf{2}})) \equiv 1_{\mathbf{2}}$$
$$\varphi(\psi(P, \mathsf{inr}(f))) \equiv (\emptyset, \mathsf{dec}\text{-}\emptyset) \qquad \psi(\varphi(0_{\mathbf{2}})) \equiv 0_{\mathbf{2}}.$$

It is therefore immediate that ψ is a retract of φ. However, in order to show that ψ is a section of φ we still need to show that

$$(\mathbf{1}, \mathsf{dec}\text{-}\mathbf{1}) = (P, \mathsf{inl}(p))$$
$$(\emptyset, \mathsf{dec}\text{-}\emptyset) = (P, \mathsf{inr}(f)).$$

Since $\mathsf{is\text{-}decidable}(P)$ is shown to be a proposition in Lemma 15.1.3, it suffices to show that

$$\mathbf{1} = P \qquad \text{if we have } p : P$$
$$\emptyset = P \qquad \text{if we have } f : \neg P.$$

In both cases we proceed by propositional extensionality. Therefore we obtain the desired identifications by observing that

$$\mathbf{1} \leftrightarrow P \qquad \text{if we have } p : P$$
$$\emptyset \leftrightarrow P \qquad \text{if we have } f : \neg P.$$
□

We will now study the concept of decidable equality.

Definition 15.1.5. We say that a type A has **decidable equality** if the identity type $x = y$ is decidable for every $x, y : A$. Types with decidable equality are also called **discrete**.

Lemma 15.1.6. *For each $m, n : \mathbb{N}$, the types $\mathsf{Eq}_{\mathbb{N}}(m, n)$, $m \leq n$ and $m < n$ are decidable.*

Proof. The proofs in each of the three cases is similar, so we only show that $\mathsf{Eq}_{\mathbb{N}}(m, n)$ is decidable for each $m, n : \mathbb{N}$. This is done by induction on m and n. Note that the types

$$\mathsf{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) \equiv \mathbf{1}$$

$$\mathsf{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}(n)) \equiv \emptyset$$
$$\mathsf{Eq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(m), 0_{\mathbb{N}}) \equiv \emptyset$$

are all decidable. Moreover, the type $\mathsf{Eq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(m), \mathsf{succ}_{\mathbb{N}}(n)) \equiv \mathsf{Eq}_{\mathbb{N}}(m, n)$ is decidable by the inductive hypothesis. □

Corollary 15.1.7. *Equality on the natural numbers is decidable.*

Proof. Recall from the proof of Theorem 10.2.6 that the canonical map

$$(m = n) \simeq \mathsf{Eq}_{\mathbb{N}}(m, n)$$

is an equivalence. Thus we obtain that $(m = n)$ is decidable from the fact that $\mathsf{Eq}_{\mathbb{N}}(m, n)$ is decidable. □

We have already shown in Theorem 10.2.6 that the type of natural numbers is a set. In fact, any type with decidable equality is a set. This fact is known as Hedberg's theorem.

Theorem 15.1.8 (Hedberg)**.** *Any type with decidable equality is a set.*

Proof. Let A be a type, and let

$$d : \prod_{(x,y:A)} (x = y) + \neg(x = y).$$

Recall from Exercise 4.2 that $(A + \neg A) \to (\neg\neg A \to A)$ for any type A, so we obtain that

$$\prod_{(x,y:A)} \neg\neg(x = y) \to (x = y).$$

Now observe that $\neg\neg(x = y)$ is a proposition for each $x, y : A$, and that the relation $x, y \mapsto \neg\neg(x = y)$ is reflexive. Therefore we are in position to apply Theorem 10.2.3 and we conclude that A is a set. □

15.2 The well-ordering principle for decidable families over $\mathbb{N}$

Definition 15.2.1. A family P over a type A is said to be decidable if $P(x)$ is decidable for every $x : A$. A **decidable subset** of a type A is a map

$$P : A \to \mathsf{classical\text{-}Prop}.$$

Definition 15.2.2. Let P be a decidable family over $\mathbb{N}$, and let $n : \mathbb{N}$ be a natural number equipped with $p : P(n)$. We say that n is a **minimal P-element** if it comes equipped with a term of type

$$\mathsf{is\text{-}minimal}_P(n, p) :\equiv \left(\prod_{(m:\mathbb{N})} P(m) \to (n \leq m)\right)$$

Note that the type $\mathsf{is\text{-}minimal}_P(n, p)$ doesn't depend on p. However, it doesn't make much sense that n is a minimal element of P unless we already know that n is in P. Indeed, if we would omit the hypothesis that n is in P, it would be more accurate to say that n is a *lower bound* of P. The following theorem is the well-ordering principle of $\mathbb{N}$.

Theorem 15.2.3. *Let P be a decidable family over $\mathbb{N}$. Then there is a function*

$$\left(\sum_{(n:\mathbb{N})} P(n)\right) \to \left(\sum_{(m:\mathbb{N})} \sum_{(p:P(m))} \mathsf{is\text{-}minimal}_P(m, p)\right).$$

Proof. Consider a universe $\mathcal{U}$ that contains P. We show by induction on $n : \mathbb{N}$ that there is a function

$$Q(n) \to \Big(\textstyle\sum_{(m:\mathbb{N})}\sum_{(p:Q(m))}\mathsf{is\text{-}minimal}_Q(m,p)\Big)$$

for every decidable family $Q : \mathbb{N} \to \mathcal{U}$. Note that we performed a swap in the order of quantification, using the universe that contains P. This slightly strengthens the inductive hypothesis, which we will be able to exploit.

The base case is trivial, since $0_{\mathbb{N}}$ is the least natural number. For the inductive step, suppose that $Q(\mathsf{succ}_{\mathbb{N}}(n))$ holds. Note that $Q(0_{\mathbb{N}})$ is assumed to be decidable, so we proceed by case analysis on $Q(0_{\mathbb{N}}) + \neg Q(0_{\mathbb{N}})$. Given $q : Q(0_{\mathbb{N}})$, it follows immediately that $0_{\mathbb{N}}$ must be minimal. In the case where $\neg Q(0_{\mathbb{N}})$, we consider the decidable subset Q' of $\mathbb{N}$ given by

$$Q'(n) :\equiv Q(\mathsf{succ}_{\mathbb{N}}(n)).$$

Since we have $q : Q'(n)$, we obtain a minimal element in Q' by the inductive hypothesis. Of course, by the assumption that $Q(0_{\mathbb{N}})$ doesn't hold, the minimal element of Q' is also the minimal element of Q. □

15.3 The strong induction principle of $\mathbb{N}$

Theorem 15.3.1. *For any type family P over $\mathbb{N}$ there an operation*

$$\mathsf{strong\text{-}ind}_{\mathbb{N}} : P(0_{\mathbb{N}}) \to \Big(\textstyle\prod_{(n:\mathbb{N})}\Big(\prod_{(m:\mathbb{N})}(m \leq n) \to P(m)\Big) \to P(n+1)\Big) \to \Big(\prod_{(n:\mathbb{N})}P(n)\Big).$$

Moreover, the operation $\mathsf{strong\text{-}ind}_{\mathbb{N}}$ *comes equipped with identifications*

$$\begin{aligned}\mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0, p_S, 0_{\mathbb{N}}) &= p_0\\ \mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0, p_S, n+1) &= p_S(n, (\lambda m.\,\lambda p.\,\mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0, p_S, m))),\end{aligned}$$

for any $p_0 : P(0_{\mathbb{N}})$ and $p_S : \prod_{(n:\mathbb{N})}\Big(\prod_{(m:\mathbb{N})}(m \leq n) \to P(m)\Big) \to P(n+1)$.

Proof. Consider

$$\begin{aligned}&p_0 : P(0_{\mathbb{N}})\\ &p_S : \textstyle\prod_{(n:\mathbb{N})}\Big(\prod_{(m:\mathbb{N})}(m \leq n) \to P(m)\Big) \to P(n+1)\end{aligned}$$

First, we claim that there is a function

$$\tilde{p}_0 : \textstyle\prod_{(m:\mathbb{N})}(m \leq 0_{\mathbb{N}}) \to P(m)$$

that comes equipped with an identification

$$\tilde{p}_0(0_{\mathbb{N}}, p) = p_0$$

for any $p : 0_{\mathbb{N}} \leq 0_{\mathbb{N}}$. The fact that we have such a dependent function $\tilde{p}_0$ follows immediately by induction on m and $p : m \leq 0_{\mathbb{N}}$.

Next, we claim that there is a function

$$\tilde{p}_S : \textstyle\prod_{(n:\mathbb{N})}\Big(\prod_{(m:\mathbb{N})}(m \leq n) \to P(m)\Big) \to \Big(\prod_{(m:\mathbb{N})}(m \leq n+1) \to P(m)\Big)$$

equipped with a homotopy

$$\prod_{(m:\mathbb{N})}\prod_{(q:m\leq n)}\prod_{(p:m\leq n+1)}\tilde{p}_S(n,H,m,p)=H(m,q)$$

and an identification

$$\tilde{p}_S(n,H,n+1,p)=p_S(n,H)$$

for every $p : n+1 \leq n+1$.

Using $\tilde{p}_0$ and $\tilde{p}_S$, we obtain by induction on n a function

$$\tilde{s} : \prod_{(n:\mathbb{N})}\prod_{(m:\mathbb{N})}(m \leq n) \to P(m)$$

satisfying the computation rules

$$\begin{aligned}\tilde{s}(0_{\mathbb{N}}) &\equiv \tilde{p}_0\\ \tilde{s}(n+1) &\equiv \tilde{p}_S(n,\tilde{s}(n)).\end{aligned}$$

Now we define

$$\mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0,p_S,n) :\equiv \tilde{s}(n,n,\mathsf{refl\text{-}leq}_{\mathbb{N}}(n)),$$

where $\mathsf{refl\text{-}leq}_{\mathbb{N}}(n) : n \leq n$ is the proof of reflexivity of $\leq$.

It remains to show that $\mathsf{strong\text{-}ind}_{\mathbb{N}}$ satisfies the identifications claimed in the statement of the theorem. The identification that computes $\mathsf{strong\text{-}ind}_{\mathbb{N}}$ at $0_{\mathbb{N}}$ is easy to obtain:

$$\begin{aligned}\mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0,p_S,0_{\mathbb{N}}) &\equiv \tilde{s}(0_{\mathbb{N}},0_{\mathbb{N}},\mathsf{refl\text{-}leq}_{\mathbb{N}}(0_{\mathbb{N}}))\\ &\equiv \tilde{p}_0(0_{\mathbb{N}},\mathsf{refl\text{-}leq}_{\mathbb{N}})\\ &= p_0.\end{aligned}$$

To construct the identification that computes $\mathsf{strong\text{-}ind}_{\mathbb{N}}$ at a successor, we start with a similar computation:

$$\begin{aligned}\mathsf{strong\text{-}ind}_{\mathbb{N}}(p_0,p_S,n+1) &\equiv \tilde{s}(n+1,n+1,\mathsf{refl\text{-}leq}_{\mathbb{N}}(n+1))\\ &\equiv \tilde{p}_S(n,\tilde{s}(n),n+1,\mathsf{refl\text{-}leq}_{\mathbb{N}}(n+1))\\ &= p_S(n,\tilde{s}(n))\end{aligned}$$

Thus we see that, in order to show that

$$p_S(n,\tilde{s}(n)) = p_S(n,(\lambda m.\,\lambda p.\,\tilde{s}(m,m,\mathsf{refl\text{-}leq}_{\mathbb{N}}(m)))),$$

we need to prove that

$$\tilde{s}(n) = \lambda m.\,\lambda p.\,\tilde{s}(m,m,\mathsf{refl\text{-}leq}_{\mathbb{N}}(m)).$$

Here we apply function extensionality, so it suffices to show that

$$\tilde{s}(n,m,p) = \tilde{s}(m,m,\mathsf{refl\text{-}leq}_{\mathbb{N}}(m))$$

for every $m : \mathbb{N}$ and $p : m \leq n$. We proceed by induction on $n : \mathbb{N}$. The base case is trivial. For the inductive step, we note that

$$\tilde{s}(n+1,m,p) = \tilde{p}_S(n,\tilde{s}(n),m,p) = \begin{cases}\tilde{s}(n,m,p) & \text{if } m \leq n\\ p_S(n,\tilde{s}(n)) & \text{if } m = n+1.\end{cases}$$

Therefore it follows by the inductive hypothesis that

$$\tilde{s}(n+1, m, p) = \tilde{s}(m, m, \mathsf{refl\text{-}leq}_{\mathbb{N}}(m))$$

if $m \leq n$ holds. In the remaining case, where $m = n+1$, note that we have

$$\begin{aligned}\tilde{s}(\mathsf{succ}_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}, \mathsf{refl\text{-}leq}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}})) &= \tilde{p}(n, \tilde{s}(n), n+1, \mathsf{refl\text{-}leq}_{\mathbb{N}}(n+1)) \\ &= p_S(n, \tilde{s}(n)).\end{aligned}$$

Therefore we see that we also have an identification

$$\tilde{s}(n+1, m, p) = \tilde{s}(m, m, \mathsf{refl\text{-}leq}_{\mathbb{N}}(m))$$

when $m = n+1$. This completes the proof of the strong induction principle for $\mathbb{N}$. □

15.4 Defining the greatest common divisor

Lemma 15.4.1. *For any $d, n : \mathbb{N}$, the type $d \mid n$ is decidable.*

Proof. We give the proof by case analysis on $(d = 0_{\mathbb{N}}) + (d \neq 0_{\mathbb{N}})$. If $d = 0_{\mathbb{N}}$, then $d \mid n$ holds if and only if $0_{\mathbb{N}} = n$, which is decidable.

If $d \neq 0_{\mathbb{N}}$, then it follows that $n \leq nd$. Therefore we obtain by the well-ordering principle of the natural numbers a minimal $m : \mathbb{N}$ that satisfies the decidable property $n \leq md$. Now we observe that $d \mid n$ holds if and only if $n = md$, which is decidable. □

Definition 15.4.2. A type family P over $\mathbb{N}$ is said to be **bounded from above** by m for some natural number m, if it comes equipped with a term of type

$$\mathsf{is\text{-}bounded}_m(P) :\equiv \prod_{(n:\mathbb{N})} P(n) \to (n \leq m).$$

Definition 15.4.3. Let P be a type family over $\mathbb{N}$, and consider $p : P(n)$. We say that n is the maximal P-number if it comes equipped with a term of type

$$\mathsf{is\text{-}maximal}_P(n, p) :\equiv \prod_{(m:\mathbb{N})} P(m) \to m \leq n.$$

In the following lemma we show that if a decidable family P is bounded from above and inhabited, then it possesses a maximal element.

Lemma 15.4.4. *Consider a decidable type family P over $\mathbb{N}$ which is bounded from above by m. Then there is a function*

$$\mathsf{maximum}_P : \Big(\sum_{(n:\mathbb{N})} P(n)\Big) \to \Big(\sum_{(n:\mathbb{N})} \sum_{(p:P(n))} \mathsf{is\text{-}maximal}_P(n, p)\Big).$$

Proof. We define the asserted function by induction on m. In the base case, if we have $p : P(n)$, then it follows from $n \leq 0$ that $n = 0$. It follows by the boundedness of P that (n, p) is maximal.

In the inductive step we proceed by case analysis on $P(\mathsf{succ}_{\mathbb{N}}(m))$. This is allowed because P is decidable. If we have $q : P(\mathsf{succ}_{\mathbb{N}}(m))$, then it follows by the boundedness of P that $(\mathsf{succ}_{\mathbb{N}}(m), q)$ is maximal. If $\neg P(\mathsf{succ}_{\mathbb{N}}(m))$, then it follows that P is bounded by m, which allows us to proceed by recursion. □

Definition 15.4.5. For any two natural numbers m, n we define the **greatest common divisor** $\gcd(m, n)$, which satisfies the following two properties:

(i) We have both $\gcd(m, n) \mid m$ and $\gcd(m, n) \mid n$.

(ii) For any $d : \mathbb{N}$ we have $d \mid \gcd(m, n)$ if and only if both $d \mid m$ and $d \mid n$ hold.

Construction. Consider the type family $P(d) :\equiv (d \mid m) \times (d \mid n)$. Then P is bounded from above by m. Moreover, $P(1)$ holds since $1 \mid n$ for any natural number n. Furthermore, the divisibility relation is decidable, so it follows that P is a family of decidable types. Now the greatest common divisor is defined as the maximal P-element, which is obtained by Lemma 15.4.4 $\square$

15.5 The Euclidean algorithm

It was immediate from our definition of the greatest common divisor of a and b that it indeed divides both a and b, and that it is the greatest such number. However, as a program that is supposed to *compute* the greatest common divisor of a and b it performs rather poorly: it checks for every n from 1 until either a or b whether it is a divisor of both a and b, and only then it gives as output the largest common divisor that it has found. In this section we give a new definition of an operation

$$\mathsf{gcd}_E : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

following Euclid's algorithm, with the opposite qualities: it will compute rather quicky a value for $\mathsf{gcd}_E(a, b)$, but it will be left as something to show that this value is indeed the greatest common divisor of a and b.

Definition 15.5.1. We define a binary operation

$$\mathsf{gcd}_E : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}).$$

Proof. We will define the operation gcd_E with the *strong* induction principle for $\mathbb{N}$, which was given as **??**. Thus it suffices to construct a function $\mathbb{N} \to \mathbb{N}$, which will provide the values for $\mathsf{gcd}_E(0_{\mathbb{N}})$, and a function

$$h_a : \left(\prod_{(x:\mathbb{N})}(x \leq a) \to \mathbb{N} \to \mathbb{N}\right) \to (\mathbb{N} \to \mathbb{N}),$$

for every $a : \mathbb{N}$, which will provide the values for $\mathsf{gcd}_E(a + 1)$.

In the base case, we simply define

$$\mathsf{gcd}_E(0_{\mathbb{N}}) :\equiv \mathsf{id}.$$

For the inductive step, consider a family of maps $F_x : \mathbb{N} \to \mathbb{N}$ indexed by $x \leq a$. We think of $F_x(b)$ as the value for $\mathsf{gcd}_E(x, b)$, so our assumption of having such a family of maps F_x is really the assumption that $\mathsf{gcd}_E(x, b)$ is defined for every $x \leq a$. Our goal is to construct a map

$$\mathsf{gcd}_E(a + 1) : \mathbb{N} \to \mathbb{N}$$

We proceed by strong induction on $b : \mathbb{N}$. In the base case, we define

$$\mathsf{gcd}_E(a + 1, 0_{\mathbb{N}}) :\equiv a + 1.$$

For the inductive step, assume that we have a number $G_y : \mathbb{N}$ for every $y \leq b$. Observe that $(b \leq a) + (a < b)$ holds for any $b : B$, see Exercise 6.3. Thus we can proceed by case analysis to define

$$h_a(F, b + 1) :\equiv \begin{cases} F_{(a+1)-(b+1)}(b + 1) & \text{if } b \leq a \\ G_{(b+1)-(a+1)} & \text{if } a < b. \end{cases}$$

This completes the inductive step, and hence we obtain a binary operation gcd_E that satisfies

$$\begin{aligned}
\mathsf{gcd}_E(0_{\mathbb{N}}, b) &\equiv b \\
\mathsf{gcd}_E(a+1, 0_{\mathbb{N}}) &\equiv a+1 \\
\mathsf{gcd}_E(a+1, b+1) &\equiv \mathsf{gcd}_E((a+1)-(b+1), b+1) && \text{if } b \leq a. \\
\mathsf{gcd}_E(succN(a), b+1) &\equiv \mathsf{gcd}_E(a+1, (b+1)-(a+1)) && \text{if } a < b.
\end{aligned}$$
□

Proposition 15.5.2. *For each $a, b : \mathbb{N}$, the number $\mathsf{gcd}_E(a, b)$ is the greatest common divisor of a and b.*

15.6 The trial division primality test

Theorem 15.6.1. *For any $n : \mathbb{N}$, the proposition $\mathsf{is\text{-}prime}(n)$ is decidable.*

It is important to note that, even when we prove that a type such as $\mathsf{is\text{-}prime}(n)$ is decidable, it is only after we *evaluate* the proof term that we know whether the type under consideration has a term or not. In other words, for any given n we don't know right away whether it is prime or not. Evaluating whether n is prime can be computationally costly, so it may be desirable in any specific situation to give a separate mathematical *argument* that decides whether or not the number is prime.

15.7 Prime decomposition

We will show now that any natural number $n > 0$ can be written as a product of primes

$$n = p_1^{k_1} \cdots p_m^{k_m}$$

This prime decomposition is unique if we require that the primes $p_i < p_{i+1}$ for each $0 < i < m$. In order to establish these facts in type theory, we first have to define finite products.

15.8 The infinitude of primes

Theorem 15.8.1. *There are infinitely many primes.*

Proof. We will show that for every $n : \mathbb{N}$ there is a prime number that is larger than n. In other words, we will construct a term of type

$$\prod_{(n:\mathbb{N})} \sum_{(p:\mathbb{N})} \mathsf{is\text{-}prime}(p) \times (n \leq p).$$

Note that the number $n! + 1$ is relatively prime to any number $m \leq n$. Therefore the primes in its prime factorization must all be larger than n. Thus, the function that assigns to n the least prime factor of $n! + 1$ shows that for any $n : \mathbb{N}$ there is a prime number p that is larger than n. □

Corollary 15.8.2. *There is a function*

$$\mathsf{prime} : \mathbb{N} \to \sum_{(p:\mathbb{N})} \mathsf{is\text{-}prime}(p)$$

that sends n to the n-th prime. This function is strictly monotone, so it is an embedding.

Exercises

15.1 Show that for any $f : \mathsf{Fin}(m) \to \mathsf{Fin}(n)$ and any $i : \mathsf{Fin}(n)$, the type $\mathsf{fib}_f(i)$ is decidable.

15.2 Consider a decidable type $P(i)$ indexed by $i : \mathsf{Fin}(n)$.

(a) Show that the type
$$\prod_{(i:\mathsf{Fin}(n))} P(i)$$
is decidable.

(b) Show that the type
$$\sum_{(i:\mathsf{Fin}(n))} P(i)$$
is decidable.

15.3 (a) Show that $\mathbb{N}$ and $\mathbf{2}$ have decidable equality. Hint: to show that $\mathbb{N}$ has decidable equality, show first that the successor function is injective.

(b) Show that if A and B have decidable equality, then so do $A + B$ and $A \times B$. Conclude that $\mathbb{Z}$ has decidable equality.

(c) Show that if A is a retract of a type B with decidable equality, then A also has decidable equality.

15.4 Define the prime-counting function $\pi : \mathbb{N} \to \mathbb{N}$.

15.5 (The Cantor-Schröder-Bernstein theorem) Let X and Y be two sets with decidable equality, and consider two maps $f : X \to Y$ and $g : Y \to X$, both of which we assume to be injective. Construct an equivalence $X \simeq Y$.

15.6 For any $k : \mathbb{Z}$, define a function $i \mapsto i + k \mod n$ of type $\mathsf{Fin}(n) \to \mathsf{Fin}(n)$. Show that this function is an equivalence.

15.7 For any $k : \mathbb{Z}$, define a function $i \mapsto i \cdot k \mod n$ of type $\mathsf{Fin}(n) \to \mathsf{Fin}(n)$. Show that this function is an equivalence if and only if $\gcd(n, k) = 1$.

15.8 Show that
$$\sum_{i=0}^{n} \binom{n-i}{i} = F_{n+1}$$

15.9 Show that if $2^n - 1$ is prime, then n is prime.

15.10 Prove Fermat's little theorem.

15.11 Extend the definition of the greatest common divisor to all integers.

15.12 Show that
$$(\mathsf{Fin}(m) \simeq \mathsf{Fin}(n)) \leftrightarrow (m = n).$$

15.13 Show that $\mathbb{N}$ satisfies **ordinal induction**, i.e., construct for any type family P over $\mathbb{N}$ a function of type
$$\mathsf{ord\text{-}ind}_{\mathbb{N}} : \Big(\prod_{(k:\mathbb{N})} \Big(\prod_{(m:\mathbb{N})} (m < k) \to P(m)\Big) \to P(k)\Big) \to \prod_{(n:\mathbb{N})} P(n).$$
Moreover, prove that
$$\mathsf{ord\text{-}ind}_{\mathbb{N}}(h, n) = h(n, \lambda m.\, \lambda p.\, \mathsf{ord\text{-}ind}_{\mathbb{N}}(h, m))$$
for any $n : \mathbb{N}$ and any $h : \prod_{(k:\mathbb{N})} \Big(\prod_{(m:\mathbb{N})} (m < k) \to P(m)\Big) \to P(k)$.

15.14 (a) Show that if A and B have decidable equality, then so do the types $A + B$ and $A \times B$.

(b) Show that $\mathbb{Z}$ and $\mathsf{Fin}(n)$ have decidable equality, for every $n : \mathbb{N}$.

15.15 Let $P : \mathbb{N} \to \mathsf{classical\text{-}Prop}$ be a decidable subset of $\mathbb{N}$.

(a) Show that $\sum_{(m:\mathbb{N})} \sum_{(p:P(m))} \mathsf{is\text{-}minimal}_P(m, p)$ is a proposition.

(b) Show that the map

$$\Big(\sum_{(n:\mathbb{N})} P(n)\Big) \to \Big(\sum_{(m:\mathbb{N})} \sum_{(p:P(m))} \mathsf{is\text{-}minimal}_P(m,p)\Big)$$

is a propositional truncation.

15.16 Suppose that $A : I \to \mathcal{U}$ is a type family over a set I with decidable equality. Show that

$$\Big(\prod_{(i:I)} \mathsf{is\text{-}contr}(A_i)\Big) \leftrightarrow \mathsf{is\text{-}contr}\Big(\prod_{(i:I)} A_i\Big).$$

16 Set theory

In this section we construct the quotient of a type by an equivalence relation. By an equivalence relation we understand a binary relation $R : A \to (A \to \mathsf{Prop})$ which is reflexive, symmetric, and transitive. In particular, we note that equivalence relations take values in Prop. The quotient A/R is constructed as the type of equivalence classes, which is just the image of the map $R : A \to (A \to \mathsf{Prop})$. Thus, our construction of the quotient by an equivalence relation is very much like the classical construction of a quotient set. Examples of set quotients are abundant. We cover two of them: the type of rational numbers and the set truncation of a type.

There is, however, a subtle issue with our construction of the set quotient as the image of the map $R : A \to (A \to \mathsf{Prop})$. What is the universe level of the quotient A/R? Let us suppose that $\mathcal{U}$ is a universe that contains A and each $R(x, y)$. Then Prop, the type of propositions in $\mathcal{U}$, is a type in the universe $\mathcal{U}^+$, constructed in Remark 6.2.1. Therefore the type Prop^A as well as the quotient A/R are also types in $\mathcal{U}^+$. That seems unfortunate, because in Zermelo-Fraenkel set theory the quotient of a set by an equivalence relation is an ordinary set, and not a more general class.

In Zermelo-Fraenkel set theory quotients are are sets because of the axiom schema of replacement. The replacement axioms assert that the image of any function is again a set. This leads us to wonder about a type theoretical variant of the replacement axioms. Indeed, there is such a variant. The type theoretic replacement property asserts that for any map $f : A \to B$ from a type A in $\mathcal{U}$ to a type B of which the *identity types* are equivalent to types in $\mathcal{U}$, the image of f is also equivalent to a type in $\mathcal{U}$, and in fact this property is a theorem. We prove it in Theorem 27.6.11, using the univalence axiom and a new construction of the image of a map.

16.1 Equivalence relations

Definition 16.1.1. Let $R : A \to (A \to \mathsf{Prop})$ be a binary relation valued in the propositions. We say that R is an **equivalence relation** if R comes equipped with

$$\begin{aligned}
&\rho : \prod_{(x:A)} R(x,x) \\
&\sigma : \prod_{(x,y:A)} R(x,y) \to R(y,x) \\
&\tau : \prod_{(x,y,z:A)} R(x,y) \to (R(y,z) \to R(x,z)),
\end{aligned}$$

witnessing that R is reflexive, symmetric, and transitive.

Definition 16.1.2. Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation. The **equivalence class** of $x : A$ is defined to be

$$[x]_R :\equiv R(x).$$

More generally, a subtype $P : A \to \mathsf{Prop}$ is said to be an **equivalence class** if it satisfies

$$\mathsf{is\text{-}equivalence\text{-}class}(P) :\equiv \exists_{(x:A)} P = R(x).$$

Furthermore, we define A/R to be the type of equivalence classes, i.e., we define

$$A/R :\equiv \sum_{(P:A\to\mathsf{Prop})} \mathsf{is\text{-}equivalence\text{-}class}(P).$$

In other words, A/R is the image of the map $[-]_R : A \to (A \to \mathsf{Prop})$. In the following proposition we characterize the identity type of A/R. As a corollary, we obtain equivalences

$$([x]_R = [y]_R) \simeq R(x,y),$$

justifying that the quotient A/R is defined to be the type of equivalence classes. Note that in our characterization of the identity type of A/R we make use of the univalence axiom.

Proposition 16.1.3. *Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation. Furthermore, consider $x : A$ and an equivalence class P. Then the canonical map*

$$([x]_R = P) \to P(x)$$

is an equivalence.

Proof. By Theorem 9.2.2 it suffices to show that the total space

$$\sum_{(P:A/R)} P(x)$$

is contractible. The center of contraction is of course $[x]_R$, which satisfies $[x]_R(x)$ by reflexivity of R. It remains to construct a contraction. Since $\sum_{(P:A/R)} P(x)$ is a subtype of A/R, we construct a contraction by showing that

$$[x]_R = P$$

whenever $P(x)$ holds. Recall that P is an equivalence relation, i.e., that there exists a $y : A$ such that $P = [y]_R$. Note that our goal is a proposition, so we may assume that we have such a y. Then we obtain that $R(x,y)$ holds from the assumption that $P(x)$ holds. Thus, we have to show that

$$[x]_R = [y]_R$$

given that $R(x,y)$ holds. By function extensionality and the univalence axiom, it is equivalent to show that

$$\prod_{(z:A)} R(x,z) \simeq R(y,z)$$

We get a function $R(x,z) \to R(y,z)$ by transitivity, since $R(y,x)$ holds by symmetry. Conversely, we get a function $R(y,z) \to R(x,z)$ directly by transitivity. Thus, we obtain that

$$R(x,z) \leftrightarrow R(y,z)$$

for any $z : A$, which is sufficient to prove that they are equivalent because R is valued in Prop. □

Corollary 16.1.4. *Consider an equivalence relation R on a type A, and let $x, y : A$. Then there is an equivalence*

$$([x]_R = [y]_R) \simeq R(x,y).$$

Proof. By Proposition 16.1.3 we have an equivalence

$$([x]_R = [y]_R) \simeq R(y,x).$$

Moreover, $R(y,x)$ is equivalent to $R(x,y)$ by symmetry of R. □

16.2 The universal property of set quotients

The quotient A/R is constructed as the image of R, so we obtain a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "q_R"] \arrow[dr, "R"'] & & A/R \arrow[dl, hook, "i_R"] \\ & \mathsf{Prop}^A, & \end{tikzcd}$$

and the embedding $i_R : A/R \to \mathsf{Prop}^A$ satisfies the universal property of the image of R. This universal property is, however, not the usual universal property of the quotient.

Definition 16.2.1. Consider a map $q : A \to B$ into a set B satisfying the property that $f(x) = f(y)$ whenever $R(x,y)$ holds. We say that q satisfies the **universal property of the set quotient by** R if for every map $f : A \to X$ into a set X such that $f(x) = f(y)$ whenever $R(x,y)$ holds, there is a unique extension

$$\begin{tikzcd} A \arrow[d, "q"'] \arrow[dr, "f"] & \\ B \arrow[r, dotted] & X. \end{tikzcd}$$

Remark 16.2.2. Formally, we express the universal property of the quotient by R as follows. Consider a map $q : A \to B$ that satisfies the property that

$$H : \prod_{(x,y:A)} R(x,y) \to (f(x) = f(y)).$$

Then there is for any set X a map

$$q^* : (B \to X) \to \Big(\sum_{(f:A\to X)} \prod_{(x,y:A)} R(x,y) \to (f(x) = f(y))\Big).$$

This map takes a function $h : B \to X$ to the pair

$$q^*(h) :\equiv (h \circ q, \lambda x.\, \lambda y.\, \lambda r.\, \mathsf{ap}_h(H_{x,y}(r))).$$

The universal property of the set quotient of R asserts that the map q^* is an equivalence for every set X. It is important to note that the universal property of set quotients is formulated with respect to sets.

Theorem 16.2.3. *Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation, and consider a map $q : A \to B$ into a set B. Then the following are equivalent.*

(i) The map q satisfies the property that

$$q(x) = q(y)$$

for every $x, y : A$ for which $R(x,y)$ holds, and moreover q satisfies the universal property of the set quotient of R.

*(ii) The map q is surjective and **effective**, which means that for each $x, y : A$ we have an equivalence*

$$(q(x) = q(y)) \simeq R(x,y).$$

(iii) *The map $R : A \to (A \to \mathsf{Prop})$ extends along q to an embedding*

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "R"'] & & B \arrow[dl, dotted, "i"] \\ & \mathsf{Prop}^A & \end{tikzcd}$$

and the embedding i satisfies the universal property of the image inclusion of R.

Proof. We first show that (ii) is equivalent to (iii), since this is the easiest part. After that, we will show that (i) is equivalent to (ii).

Assume that (ii) holds. Then q is surjective by Theorem 14.2.4. Moreover, we have

$$\begin{aligned} R(x,y) &\simeq R(x) = R(y) \\ &\simeq i(q(x)) = i(q(y)) \\ &\simeq q(x) = q(y) \end{aligned}$$

In this calculation, the first equivalence holds by Corollary 16.1.4; the second equivalence holds since we have a homotopy $R \sim i \circ q$; and the third equivalence holds since i is an embedding. This completes the proof that (ii) implies (iii).

Next, we show that (iii) implies (ii). Assuming (iii), we define a map

$$i : B \to \mathsf{Prop}^A$$

by $i(b,a) :\equiv b = q(a)$. Then the triangle

$$\begin{tikzcd} A \arrow[rr, "q"] \arrow[dr, "R"'] & & B \arrow[dl, "i"] \\ & \mathsf{Prop}^A & \end{tikzcd}$$

commutes, since we have an equivalence

$$i(q(a), a') \simeq R(a, a')$$

for each $a, a' : A$. To show that i is an embedding, it suffices to show that i is injective, i.e., that

$$\prod_{(b,b':B)} (i(b) = i(b')) \to (b = b')$$

Note that this is a property, and that q is assumed to be surjective. Hence by Proposition 14.2.3 it is equivalent to show that

$$\prod_{(a,a':A)} (i(q(a)) = i(q(a'))) \to (q(a) = q(a')).$$

Since $R \sim i \circ q$, and $q(a) = q(a')$ is assumed to be equivalent to $R(a, a')$, it suffices to show that

$$\prod_{(a,a':A)} (R(a) = R(a')) \to R(a, a'),$$

which follows directly from Corollary 16.1.4. Thus we have shown that the factorization $R \sim i \circ q$ factors R as a surjective map followed by an injective map. We conclude by Theorem 14.2.4 that the embedding i satisfies the universal property of the image factorization of R, which finishes the proof that (iii) implies (ii).

Now we show that (i) implies (ii). To see that q is surjective if it satisfies the assumptions in (i), consider the image factorization

$$\begin{tikzcd} A \arrow[rr, "q_q"] \arrow[dr, "q"'] & & \mathrm{im}(q) \arrow[dl, "i_q"] \\ & B. & \end{tikzcd}$$

We claim that the map i_q has a section. To see this, we first note that we have

$$q_q(x) = q_q(y)$$

for any $x, y : A$ satisfying $R(x, y)$, because if $R(x, y)$ holds, then $q(x) = q(y)$ and hence $i_q(q_q(x)) = i_q(q_q(y))$ holds and i_q is an embedding. Since $\mathrm{im}(q)$ is a set, we may apply the universal property of q and we obtain a unique extension of q_q along q

$$\begin{tikzcd} A \arrow[d, "q"'] \arrow[dr, "q_q"] & \\ B \arrow[r, dotted, "h"'] & \mathrm{im}(q). \end{tikzcd}$$

Now we observe that the composite $i_q \circ h$ is an extension of q along q, so it must be the identity function by uniqueness. Thus we have established that h is a section of i_q. Now it follows from the fact that i_q is an embedding with a section, that i_q is an equivalence. We conclude that q is surjective, because q is the composite $i_q \circ q_q$ of a surjective map followed by an equivalence.

Now we have to show that $q(x) = q(y)$ is equivalent to $R(x, y)$. We first apply the universal property of q to obtain for each $x : A$ an extension of $R(x)$ along q

$$\begin{tikzcd} A \arrow[d, "q"'] \arrow[dr, "R(x)"] & \\ B \arrow[r, dotted, "\tilde{R}(x)"'] & \mathsf{Prop}. \end{tikzcd}$$

Since the triangle commutes, we have an equivalence $\tilde{R}(x, q(x')) \simeq R(x, x')$ for each $x' : A$. Now we apply Theorem 9.2.2 to see that the canonical family of maps

$$\prod_{(y:B)}(q(x) = y) \to \tilde{R}(x, y)$$

is a family of equivalences. Thus, we need to show that the type $\sum_{(y:B)} \tilde{R}(x, y)$ is contractible. For the center of contraction, note that we have $q(x) : B$, and the type $\tilde{R}(x, q(x))$ is equivalent to the type $R(x, x)$, which is inhabited by reflexivity of R. To construct the contraction, it suffices to show that

$$\prod_{(y:B)}\tilde{R}(x, y) \to (q(x) = y).$$

Since this is a property, and since we have already shown that q is a surjective map, we may apply Proposition 14.2.3, by which it suffices to show that

$$\prod_{(x':A)}\tilde{R}(x, q(x')) \to (q(x) = q(x')).$$

Since $\tilde{R}(x, q(x')) \simeq R(x, x')$, this is immediate from our assumption on q. Thus we obtain the contraction, and we conclude that we have an equivalence $\tilde{R}(x, y) \simeq (q(x) = y)$ for each $y : B$. It follows that we have an equivalence

$$R(x, y) \simeq (q(x) = q(y))$$

for each $x, y : A$, which completes the proof that (i) implies (ii).

It remains to show that (iii) implies (i). Assume (iii), and let $f : A \to X$ be a map into a set X, satisfying the property that

$$\prod_{(a,a':A)} R(a, a') \to (f(a) = f(a')).$$

Our goal is to show that the type of extensions of f along q is contractible. By Exercise 14.4 it follows that there is at most one such an extension, so it suffices to construct one.

In order to construct an extension, we will construct for every $b : B$ a term $x : X$ satisfying the property

$$P(x) :\equiv \exists_{(a:A)}(f(a) = x) \wedge (q(a) = b).$$

Before we make this construction, we first observe that there is at most one such x, i.e., that the type of $x : X$ satisfying $P(x)$ is in fact a proposition. To see this, we need to show that $x = x'$ for any $x, x' : X$ satisfying $P(x)$ and $P(x')$. Since X is assumed to be a set, our goal of showing that $x = x'$ is a property. Therefore we may assume that we have $a, a' : A$ satisfying

$$\begin{aligned} f(a) &= x & \qquad q(a) &= b \\ f(a') &= x' & \qquad q(a') &= b. \end{aligned}$$

It follows from these assumptions that $q(a) = q(a')$, and hence that $R(a, a')$ holds. This in turn implies that $f(a) = f(a')$, and hence that $x = x'$.

Now let $b : B$. Our goal is to construct an $x : X$ that satisfies the property

$$\exists_{(a:A)}(f(a) = x) \wedge (q(a) = b).$$

Since q is assumed to be surjective, we have $\|\mathsf{fib}_q(b)\|$. Moreover, since we have shown that at most one $x : X$ exists with the asserted property, we get to assume that we have $a : A$ satisfying $q(a) = b$. Now we see that $x :\equiv f(a)$ satisfies the desired property.

Thus, we obtain a function $h : B \to X$ satisfying the property that for all $b : B$ there exists an $a : A$ such that

$$f(a) = h(b) \qquad \text{and} \qquad q(a) = b.$$

In particular, it follows that $h(q(a)) = f(a)$ for all $a : A$, which completes the proof that (ii) implies (i). □

16.3 The rational numbers

16.4 Set truncation

Lemma 16.4.1. *For each type A, the relation $I_{(-1)} : A \to (A \to \mathsf{Prop})$ given by*

$$I_{(-1)}(x, y) :\equiv \|x = y\|$$

is an equivalence relation.

Proof. For every $x : A$ we have $|\mathsf{refl}_x| : \|x = x\|$, so the relation is reflexive. To see that the relation is symmetric note that by the universal property of propositional truncation there is a unique map $\|\mathsf{inv}\| : \|x = y\| \to \|y = x\|$ for which the square

$$\begin{tikzcd}
(x = y) \arrow[r, "\mathsf{inv}"] \arrow[d, "|-|"'] & (y = x) \arrow[d, "|-|"] \\
\|x = y\| \arrow[r, dotted, "\|\mathsf{inv}\|"'] & \|y = x\|
\end{tikzcd}$$

commutes. This shows that the relation is symmetric. Similarly, we show by the universal property of propositional truncation that the relation is transitive. □

Definition 16.4.2. For each type A we define the **set truncation**

$$\|A\|_0 :\equiv A/I_{(-1)},$$

and the unit of the set truncation is defined to be the quotient map.

Theorem 16.4.3. *For each type A, the set truncation satisfies the universal property of the set truncation.*

Exercises

16.1 Consider a map $f : A \to B$ into a set B, and let $R : A \to (A \to \mathsf{Prop})$ be the equivalence relation given by

$$R(x, y) :\equiv f(x) = f(y).$$

Show that the map $q_f : A \to \mathrm{im}(f)$ satisfies the universal property of the set quotient of R.

16.2 Show that the set truncation of a loop space is a group.

16.3 Recall that a normal subgroup H of a group G is a subgroup of G such that ghg^{-1} is in H for every $h : H$ and $g : G$. Given a normal subgroup H of G, we write G/H for the quotient of G by the equivalence relation where $g \sim g'$ if and only if there is a $h : H$ such that $gh = g'$. Show that G/H is again a group.

16.4 (a) Show that any proposition is locally small.
(b) Show that any essentially small type is locally small.
(c) Show that the function type $A \to X$ is locally small whenever A is essentially small and X is locally small.

16.5 Let $f : A \to B$ be a map. Show that the following are equivalent:

(i) The map f is **locally small** in the sense that for every $x, y : A$, the action on paths of f

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

is an essentially small map.

(ii) The diagonal δ_f of f as defined in Exercise 21.2 is classified by the universal fibration.

16.6 Use Theorem 11.2.1, **??**, and Corollary 11.3.2 to show that the type

$$\mathsf{span}(A, B) :\equiv \textstyle\sum_{(S:\mathcal{U})} (S \to A) \times (S \to B)$$

of small spans from A to B is equivalent to the type $A \to (B \to \mathcal{U})$ of small relations from A to B.

17 Groups in univalent mathematics

In this section we demonstrate a typical way to use the univalence axiom, showing that isomorphic groups can be identified. This is an instance of the *structure identity principle*, which is described in more detail in section 9.8 of [3]. We will see that in order to establish the fact that isomorphic groups can be identified, it has to be part of the definition of a group that its underlying type is a set. This is an important observation: in many branches of algebra the objects of study are *set-level* structures[1].

17.1 Semi-groups and groups

We introduce the type of groups in two stages: first we introduce the type of *semi-groups*, and then we introduce groups as semi-groups that possess further structure. It will turn out that this further structure is in fact a property, and this fact will help us to prove that isomorphic groups are equal.

Definition 17.1.1. A **semi-group** consists of a set G equipped with a term of type $\mathsf{has\text{-}associative\text{-}mul}(G)$, which is the type of pairs $(\mu_G, \mathsf{assoc}_G)$ consisting of a binary operation

$$\mu_G : G \to (G \to G)$$

and a homotopy

$$\mathsf{assoc}_G : \prod_{(x,y,z:G)} \mu_G(\mu_G(x,y),z) = \mu_G(x,\mu_G(y,z)).$$

We write $\mathsf{Semi\text{-}Group}$ for the type of all semi-groups in $\mathcal{U}$.

Definition 17.1.2. A semi-group G is said to be **unital** if it comes equipped with a **unit** $e_G : G$ that satisfies the left and right unit laws

$$\mathsf{left\text{-}unit}_G : \prod_{(y:G)} \mu_G(e_G, y) = y$$
$$\mathsf{right\text{-}unit}_G : \prod_{(x:G)} \mu_G(x, e_G) = x.$$

We write $\mathsf{is\text{-}unital}(G)$ for the type of such triples $(e_G, \mathsf{left\text{-}unit}_G, \mathsf{right\text{-}unit}_G)$. Unital semi-groups are also called **monoids**.

The unit of a semi-group is of course unique once it exists. In univalent mathematics we express this fact by asserting that the type $\mathsf{is\text{-}unital}(G)$ is a proposition for each semi-group G. In other words, being unital is a *property* of semi-groups rather than structure on it. This is typical for univalent mathematics: we express that a structure is a property by proving that this structure is a proposition.

Lemma 17.1.3. *For a semi-group G the type $\mathsf{is\text{-}unital}(G)$ is a proposition.*

Proof. Let G be a semi-group. Note that since G is a set, it follows that the types of the left and right unit laws are propositions. Therefore it suffices to show that any two terms $e, e' : G$ satisfying the left and right unit laws can be identified. This is easy:

$$e = \mu_G(e, e') = e'. \qquad \square$$

[1] A notable exception is that of categories, which are objects at truncation level 1, i.e., at the level of *groupoids*. We will briefly introduce categories in §17.5. For more about categories we recommend Chapter 9 of [3].

Definition 17.1.4. Let G be a unital semi-group. We say that G **has inverses** if it comes equipped with an operation $x \mapsto x^{-1}$ of type $G \to G$, satisfying the left and right inverse laws

$$\begin{aligned}\mathsf{left\text{-}inv}_G &: \textstyle\prod_{(x:G)} \mu_G(x^{-1}, x) = e_G \\ \mathsf{right\text{-}inv}_G &: \textstyle\prod_{(x:G)} \mu_G(x, x^{-1}) = e_G.\end{aligned}$$

We write $\mathsf{is\text{-}group}'(G, e)$ for the type of such triples $((-)^{-1}, \mathsf{left\text{-}inv}_G, \mathsf{right\text{-}inv}_G)$, and we write

$$\mathsf{is\text{-}group}(G) :\equiv \textstyle\sum_{(e:\mathsf{is\text{-}unital}(G))} \mathsf{is\text{-}group}'(G, e)$$

A **group** is a unital semi-group with inverses. We write Group for the type of all groups in $\mathcal{U}$.

Lemma 17.1.5. *For any semi-group G the type* $\mathsf{is\text{-}group}(G)$ *is a proposition.*

Proof. We have already seen that the type $\mathsf{is\text{-}unital}(G)$ is a proposition. Therefore it suffices to show that the type $\mathsf{is\text{-}group}'(G, e)$ is a proposition for any $e : \mathsf{is\text{-}unital}(G)$.

Since a semi-group G is assumed to be a set, we note that the types of the inverse laws are propositions. Therefore it suffices to show that any two inverse operations satisfying the inverse laws are homotopic.

Let $x \mapsto x^{-1}$ and $x \mapsto \bar{x}^{-1}$ be two inverse operations on a unital semi-group G, both satisfying the inverse laws. Then we have the following identifications

$$\begin{aligned}x^{-1} &= \mu_G(e_G, x^{-1}) \\ &= \mu_G(\mu_G(\bar{x}^{-1}, x), x^{-1}) \\ &= \mu_G(\bar{x}^{-1}, \mu_G(x, x^{-1})) \\ &= \mu_G(\bar{x}^{-1}, e_G) \\ &= \bar{x}^{-1}\end{aligned}$$

for any $x : G$. Thus the two inverses of x are the same, so the claim follows. □

Example 17.1.6. An important class of examples consists of **loop spaces** $x = x$ of a 1-type X, for any $x : X$. We will write $\Omega(X, x)$ for the loop space of X at x. Since X is assumed to be a 1-type, it follows that the type $\Omega(X, x)$ is a set. Then we have

$$\begin{aligned}\mathsf{refl}_x &: \Omega(X, x) \\ \mathsf{inv} &: \Omega(X, x) \to \Omega(X, x) \\ \mathsf{concat} &: \Omega(X, x) \to (\Omega(X, x) \to \Omega(X, x)),\end{aligned}$$

and these operations satisfy the group laws, since the group laws are just a special case of the groupoid laws for identity types, constructed in §5.2.

Example 17.1.7. The type $\mathbb{Z}$ of integers can be given the structure of a group, with the group operation being addition. The fact that $\mathbb{Z}$ is a set follows from Theorem 10.2.6 and Exercise 10.5. The group laws were shown in Exercise 7.9.

Example 17.1.8. Our last class of examples consists of the **automorphism groups** on sets. Given a set X, we define

$$\mathrm{Aut}(X) :\equiv (X \simeq X).$$

The group operation of $\mathrm{Aut}(X)$ is just composition of equivalences, and the unit of the group is the identity function. Note however, that although function composition is strictly associative and satisfies the unit laws strictly, composition of equivalences only satisfies the group laws up to identification because the proof that composites are equivalences is carried along.

Important special cases of the automorphism groups are the **symmetric groups**

$$\mathcal{S}_n :\equiv \mathsf{Aut}(\mathsf{Fin}(n)).$$

17.2 Homomorphisms of semi-groups and groups

Definition 17.2.1. Let G and H be semi-groups. A **homomorphism** of semi-groups from G to H is a pair (f, μ_f) consisting of a function $f : G \to H$ between their underlying types, and a term

$$\mu_f : \prod_{(x,y:G)} f(\mu_G(x,y)) = \mu_H(f(x), f(y))$$

witnessing that f preserves the binary operation of G. We will write

$$\mathsf{hom}(G, H)$$

for the type of all semi-group homomorphisms from G to H.

Remark 17.2.2. Since it is a property for a function to preserve the multiplication of a semi-group, it follows easily that equality of semi-group homomorphisms is equivalent to the type of homotopies between their underlying functions. In particular, it follows that the type of homomorphisms of semi-groups is a set.

Remark 17.2.3. The **identity homomorphism** on a semi-group G is defined to be the pair consisting of

$$\mathsf{id} : G \to G$$

$$\lambda x.\, \lambda y.\, \mathsf{refl}_{xy} : \prod_{(x,y:G)} xy = xy.$$

Let $f : G \to H$ and $g : H \to K$ be semi-group homomorphisms. Then the composite function $g \circ f : G \to K$ is also a semi-group homomorphism, since we have the identifications

$$g(f(xy)) = g(f(x)f(y)) = g(f(x))g(f(y)).$$

Since the identity type of semi-group homomorphisms is equivalent to the type of homotopies between semi-group homomorphisms it is easy to see that semi-group homomorphisms satisfy the laws of a category, i.e., that we have the identifications

$$\mathsf{id} \circ f = f$$

$$g \circ \mathsf{id} = g$$

$$(h \circ g) \circ f = h \circ (g \circ f)$$

for any composable semi-group homomorphisms f, g, and h. Note, however that these equalities are not expected to hold judgmentally, since preservation of the semi-group operation is part of the data of a semi-group homomorphism.

Definition 17.2.4. Let G and H be groups. A **homomorphism** of groups from G to H is defined to be a semi-group homomorphism between their underlying semi-groups. We will write

$$\mathsf{hom}(G, H)$$

for the type of all group homomorphisms from G to H.

Remark 17.2.5. Since a group homomorphism is just a semi-group homomorphism between the underlying semi-groups, we immediately obtain the identity homomorphism, composition, and the category laws are satisfied.

17.3 Isomorphic semi-groups are equal

Definition 17.3.1. Let $h : \mathsf{hom}(G, H)$ be a homomorphism of semi-groups. Then h is said to be an **isomorphism** if it comes equipped with a term of type $\mathsf{is\text{-}iso}(h)$, consisting of triples (h^{-1}, p, q) consisting of a homomorphism $h^{-1} : \mathsf{hom}(H, G)$ of semi-groups and identifications

$$p : h^{-1} \circ h = \mathsf{id}_G \qquad \text{and} \qquad q : h \circ h^{-1} = \mathsf{id}_H$$

witnessing that h^{-1} satisfies the inverse lawsWe write $G \cong H$ for the type of all isomorphisms of semi-groups from G to H, i.e.,

$$G \cong H :\equiv \sum_{(h:\mathsf{hom}(G,H))} \sum_{(k:\mathsf{hom}(H,G))} (k \circ h = \mathsf{id}_G) \times (h \circ k = \mathsf{id}_H).$$

If f is an isomorphism, then its inverse is unique. In other words, being an isomorphism is a property.

Lemma 17.3.2. *For any semi-group homomorphism $h : \mathsf{hom}(G, H)$, the type*

$$\mathsf{is\text{-}iso}(h)$$

is a proposition. It follows that the type $G \cong H$ is a set for any two semi-groups G and H.

Proof. Let k and k' be two inverses of h. In Remark 17.2.2 we have observed that the type of semi-group homomorphisms between any two semi-groups is a set. Therefore it follows that the types $h \circ k = \mathsf{id}$ and $k \circ h = \mathsf{id}$ are propositions, so it suffices to check that $k = k'$. In Remark 17.2.2 we also observed that the equality type $k = k'$ is equivalent to the type of homotopies $k \sim k'$ between their underlying functions. We construct a homotopy $k \sim k'$ by the usual argument:

$$k(y) = k(h(k'(y))) = k'(y). \qquad \square$$

Lemma 17.3.3. *A semi-group homomorphism $h : \mathsf{hom}(G, H)$ is an isomorphism if and only if its underlying map is an equivalence. Consequently, there is an equivalence*

$$(G \cong H) \simeq \sum_{(e:G \simeq H)} \prod_{(x,y:G)} e(\mu_G(x, y)) = \mu_H(e(x), e(y))$$

Proof. If $h : \mathsf{hom}(G, H)$ is an isomorphism, then the inverse semi-group homomorphism also provides an inverse of the underlying map of h. Thus we obtain that h is an equivalence. The standard proof showing that if the underlying map $f : G \to H$ of a group homomorphism is invertible then its inverse is again a group homomorphism also works in type theory. $\square$

Definition 17.3.4. Let G and H be a semi-groups. We define the map

$$\mathsf{iso\text{-}eq} : (G = H) \to (G \cong H)$$

by path induction, taking refl_G to isomorphism id_G.

Theorem 17.3.5. *The map*

$$\mathsf{iso\text{-}eq} : (G = H) \to (G \cong H)$$

is an equivalence for any two semi-groups G and H.

Proof. By the fundamental theorem of identity types Theorem 9.2.2 it suffices to show that the total space

$$\textstyle\sum_{(G':\mathsf{Semi\text{-}Group})} G \cong G'$$

is contractible. Since the type of isomorphisms from G to G' is equivalent to the type of equivalences from G to G' it suffices to show that the type

$$\textstyle\sum_{(G':\mathsf{Semi\text{-}Group})}\sum_{(e:G\simeq G')}\prod_{(x,y:G)} e(\mu_G(x,y)) = \mu_{G'}(e(x),e(y)))$$

is contractible[2]. Since $\mathsf{Semi\text{-}Group}$ is the Σ-type

$$\textstyle\sum_{(G':\mathsf{Set})} \mathsf{has\text{-}associative\text{-}mul}(G'),$$

it suffices to show that the types

$$\textstyle\sum_{(G':\mathsf{Set})} G \simeq G'$$

$$\textstyle\sum_{(\mu':\mathsf{has\text{-}associative\text{-}mul}(G))}\prod_{(x,y:G)} \mu_G(x,y) = \mu'(x,y)$$

is contractible. The first type is contractible by the univalence axiom. The second type is contractible by function extensionality. ☐

Corollary 17.3.6. *The type* $\mathsf{Semi\text{-}Group}$ *is a* 1*-type.*

Proof. It is straightforward to see that the type of group isomorphisms $G \cong H$ is a set, for any two groups G and H. ☐

17.4 Isomorphic groups are equal

Analogously to the map $\mathsf{iso\text{-}eq}$ of semi-groups, we have a map $\mathsf{iso\text{-}eq}$ of groups. Note, however, that the domain of this map is now the identity type $G = H$ of the *groups* G and H, so the maps $\mathsf{iso\text{-}eq}$ of semi-groups and groups are not exactly the same maps.

Definition 17.4.1. Let G and H be groups. We define the map

$$\mathsf{iso\text{-}eq} : (G = H) \to (G \cong H)$$

by path induction, taking refl_G to the identity isomorphism $\mathsf{id} : G \cong G$.

[2] In order to show that a type of the form

$$\textstyle\sum_{((x,y):\sum_{(x:A)} B(x))}\sum_{(z:C(x))} D(x,y,z)$$

is contractible, a useful strategy is to first show that the type $\sum_{(x:A)} C(x)$ is contractible. Once this is established, say with center of contraction (x_0, z_0), it suffices to show that the type $\sum_{(y:B(x_0))} D(x_0, y, z_0)$ is contractible.

Theorem 17.4.2. *For any two groups G and H, the map*

$$\mathsf{iso\text{-}eq} : (G = H) \to (G \cong H)$$

is an equivalence.

Proof. Let G and H be groups, and write UG and UH for their underlying semi-groups, respectively. Then we have a commuting triangle

$$\begin{tikzcd}
(G = H) \arrow[rr, "\mathsf{ap}_{\mathsf{pr}_1}"] \arrow[dr, "\mathsf{iso\text{-}eq}"'] & & (UG = UH) \arrow[dl, "\mathsf{iso\text{-}eq}"] \\
& (G \cong H) &
\end{tikzcd}$$

Since being a group is a property of semi-groups it follows that the projection map $\mathsf{Group} \to \mathsf{Semi\text{-}Group}$ forgetting the unit and inverses, is an embedding. Thus the top map in this triangle is an equivalence. The map on the right is an equivalence by Theorem 17.3.5, so the claim follows by the 3-for-2 property. □

Corollary 17.4.3. *The type of groups is a* 1*-type.*

17.5 Categories in univalent mathematics

In our proof of the fact that isomorphic groups are equal we have made extensive use of the notion of group homomorphism. What we have shown, in fact, is that there is a category of groups which is *Rezk complete* in the sense that the type of isomorphisms between two objects is equivalent to the type of identifications between those objects. In this final section we briefly introduce the notion of Rezk complete category. There are many more examples of categories, such as the categories of rings, or modules over a ring.

Definition 17.5.1. A **pre-category** $\mathcal{C}$ consists of

(i) A type A of **objects**.

(ii) For every two objects $x, y : A$ a set

$$\mathsf{hom}(x, y)$$

of **morphisms** from x to y.

(iii) For every object $x : A$ an **identity morphism**

$$\mathsf{id} : \mathsf{hom}(x, x)$$

(iv) For every two morphisms $f : \mathsf{hom}(x, y)$ and $g : \mathsf{hom}(y, z)$, a morphism

$$g \circ f : \mathsf{hom}(x, z)$$

called the **composition** of f and g.

(v) the following terms

$$\begin{aligned}\mathsf{left\text{-}unit}_{\mathcal{C}} &: \mathsf{id} \circ f = f \\ \mathsf{right\text{-}unit}_{\mathcal{C}} &: g \circ \mathsf{id} = g \\ \mathsf{assoc}_{\mathcal{C}} &: (h \circ g) \circ f = h \circ (g \circ f)\end{aligned}$$

witnessing that the category laws are satisfied.

Example 17.5.2. Since the type $X \to Y$ of functions between sets is again a set, we have a pre-category of sets.

Example 17.5.3. By Remarks 17.2.3 and 17.2.5 we have pre-categories of semi-groups and of groups.

Example 17.5.4. A pre-category satisfying the condition that every hom-set is a proposition is a **preorder**.

Definition 17.5.5. Given a pre-category $\mathcal{C}$, a morphism $f : \mathsf{hom}(x, y)$ is said to be an **isomorphism** if there exists a morphism $g : \mathsf{hom}(y, x)$ such that

$$\begin{aligned}g \circ f &= \mathsf{id} \\ f \circ g &= \mathsf{id}.\end{aligned}$$

We will write $\mathsf{iso}(x, y)$ for the type of all isomorphisms in $\mathcal{C}$ from x to y.

Remark 17.5.6. Just as in the case for semi-groups and groups, the condition that $f : \mathsf{hom}(x, y)$ is an isomorphism is a property of f.

Definition 17.5.7. A pre-category $\mathcal{C}$ is said to be **Rezk-complete** if the canonical map

$$(x = y) \to \mathsf{iso}(x, y)$$

is an equivalence for any two objects x and y of $\mathcal{C}$. Rezk-complete pre-categories are also called **categories**.

Example 17.5.8. The pre-category of sets is Rezk complete by the univalence axiom, so it is a category.

Example 17.5.9. The pre-categories of semi-groups and groups are Rezk-complete. Therefore they form categories.

Example 17.5.10. A pre-order is Rezk-complete if and only if it is anti-symmetric. In other words, a poset is precisely a category for which all the hom-sets are propositions. Thus, we see that the anti-symmetry axiom can be seen as a univalence axiom for pre-orders.

Exercises

17.1 Let X be a set. Show that the map

$$\mathsf{equiv\text{-}eq} : (X = X) \to (X \simeq X)$$

is a group isomorphism.

17.2 (a) Consider a group G. Show that the function

$$\mu_G : G \to (G \simeq G)$$

is an injective group homomorphism.

(b) Consider a pointed type A. Show that the concatenation function

$$\mathsf{concat} : \Omega(A) \to (\Omega(A) \simeq \Omega(A))$$

is an embedding.

17.3 Let $f : \mathsf{hom}(G, H)$ be a group homomorphism. Show that f preserves units and inverses, i.e., show that

$$f(e_G) = e_H$$
$$f(x^{-1}) = f(x)^{-1}.$$

17.4 Give a direct proof and a proof using the univalence axiom of the fact that all semi-group isomorphisms between unital semi-groups preserve the unit. Conclude that isomorphic monoids are equal.

17.5 Consider a monoid M with multiplication $\mu : M \to (M \to M)$ and unit e. Write

$$\bar{\mu} :\equiv \mathsf{fold\text{-}list}(e, \mu) : \mathsf{list}(M) \to M$$

for the iterated multiplication operation (see Exercise 4.8). Show that the square

$$\begin{tikzcd}
\mathsf{list}(\mathsf{list}(M)) \arrow[r, "\mathsf{flatten\text{-}list}(M)"] \arrow[d, "\mathsf{list}(\bar{\mu})"'] & \mathsf{list}(M) \arrow[d, "\bar{\mu}"] \\
\mathsf{list}(M) \arrow[r, "\bar{\mu}"'] & M
\end{tikzcd}$$

commutes.

17.6 Construct the category of posets.

17.7 Consider the **walking isomorphism**, i.e., the pre-category $\mathcal{I}$ given by

$$\begin{tikzcd}
0 \arrow[r, bend left, "f"] & 1 \arrow[l, bend left, "f^{-1}"]
\end{tikzcd}$$

satisfying $f \circ f^{-1} = \mathsf{id}$ and $f^{-1} \circ f = \mathsf{id}$. Show that for any pre-category $\mathcal{C}$ the following are equivalent:

(i) The pre-category $\mathcal{C}$ is Rezk complete.

(ii) The precomposition function

$$\mathcal{C} \to \mathsf{Fun}(\mathcal{I}, \mathcal{C})$$

is an equivalence.

18 The circle

We have seen inductive types, in which we describe a type by its constructors and an induction principle that allows us to construct sections of dependent types. Inductive types are freely generated by their constructors, which describe how we can construct their terms.

However, many familiar constructions in algebra involve the construction of algebras by generators and relations. For example, the free abelian group with two generators is described as the group with generators x and y, and the relation $xy = yx$.

In this chapter we introduce higher inductive types, where we follow a similar idea: to allow in the specification of inductive types not only *point constructors*, but also *path constructors* that give us relations between the point constructors. The ideas behind the definition of higher inductive types are introduced by studying the simplest non-trivial example: the *circle*.

18.1 The induction principle of the circle

The *circle* is defined as a higher inductive type $\mathbf{S}^1$ that comes equipped with

$$\begin{aligned} &\mathsf{base} : \mathbf{S}^1 \\ &\mathsf{loop} : \mathsf{base} = \mathsf{base}. \end{aligned}$$

Just like for ordinary inductive types, the induction principle for higher inductive types provides us with a way of constructing sections of dependent types. However, we need to take the *path constructor* loop into account in the induction principle.

By applying a section $f : \prod_{(x:\mathbf{S}^1)} P(x)$ to the base point of the circle, we obtain a term $f(\mathsf{base}) : P(\mathsf{base})$. Moreover, using the dependent action on paths of f of Definition 5.4.2 we also obtain for any dependent function $f : \prod_{(x:\mathbf{S}^1)} P(x)$ a path

$$\mathsf{apd}_f(\mathsf{loop}) : \mathsf{tr}_P(\mathsf{loop}, f(\mathsf{base})) = f(\mathsf{base})$$

in the fiber $P(\mathsf{base})$.

Definition 18.1.1. Let P be a type family over the circle. The **dependent action on generators** is the map

$$\mathsf{dgen}_{\mathbf{S}^1} : \Big(\prod_{(x:\mathbf{S}^1)} P(x)\Big) \to \Big(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\Big) \tag{18.1}$$

given by $\mathsf{dgen}_{\mathbf{S}^1}(f) :\equiv (f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop}))$.

We now give the full specification of the circle.

Definition 18.1.2. The **circle** is a type $\mathbf{S}^1$ that comes equipped with

$$\begin{aligned} &\mathsf{base} : \mathbf{S}^1 \\ &\mathsf{loop} : \mathsf{base} = \mathsf{base}, \end{aligned}$$

and satisfies the **induction principle of the circle**, which provides for each type family P over $\mathbf{S}^1$ a map

$$\mathsf{ind}_{\mathbf{S}^1} : \Big(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\Big) \to \Big(\prod_{(x:\mathbf{S}^1)} P(x)\Big),$$

and a homotopy witnessing that $\mathsf{ind}_{\mathbf{S}^1}$ is a section of $\mathsf{dgen}_{\mathbf{S}^1}$

$$\mathsf{comp}_{\mathbf{S}^1} : \mathsf{dgen}_{\mathbf{S}^1} \circ \mathsf{ind}_{\mathbf{S}^1} \sim \mathsf{id}$$

for the computation rule.

Remark 18.1.3. The type of identifications $(y, p) = (y', p')$ in the type

$$\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y$$

is equivalent to the type of pairs (α, β) consisting of an identification $\alpha : y = y'$, and an identification β witnessing that the square

$$\begin{tikzcd}
\mathsf{tr}_P(\mathsf{loop}, y) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "p"'] & \mathsf{tr}_P(\mathsf{loop}, y') \arrow[d, equal, "p'"] \\
y \arrow[r, equal, "\alpha"'] & y'
\end{tikzcd}$$

commutes. Therefore it follows from the induction principle of the circle that for any $(y, p) : \sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y$, there is a dependent function $f : \prod_{(x:\mathbf{S}^1)} P(x)$ equipped with an identification

$$\alpha : f(\mathsf{base}) = y,$$

and an identification β witnessing that the square

$$\begin{tikzcd}
\mathsf{tr}_P(\mathsf{loop}, f(\mathsf{base})) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "\mathsf{apd}_f(\mathsf{loop})"'] & \mathsf{tr}_P(\mathsf{loop}, y) \arrow[d, equal, "p"] \\
f(\mathsf{base}) \arrow[r, equal, "\alpha"'] & y
\end{tikzcd}$$

commutes.

18.2 The (dependent) universal property of the circle

Our goal is now to use the induction principle of the circle to derive the **universal property** of the circle. This universal property states that, for any type X the canonical map

$$\left(\mathbf{S}^1 \to X\right) \to \left(\sum_{(x:X)} x = x\right)$$

given by $f \mapsto (f(\mathsf{base}), \mathsf{ap}_f(\mathsf{loop}))$ is an equivalence. It turns out that it is easier to prove the **dependent universal property** first. The dependent universal property states that for any type family P over the circle, the canonical map

$$\left(\prod_{(x:\mathbf{S}^1)} P(x)\right) \to \left(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\right)$$

given by $f \mapsto (f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop}))$ is an equivalence.

Theorem 18.2.1. *For any type family P over the circle, the map*

$$\left(\prod_{(x:\mathbf{S}^1)} P(x)\right) \to \left(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\right)$$

given by $f \mapsto (f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop}))$ *is an equivalence.*

Proof. By the induction principle of the circle we know that the map has a section, i.e., we have

$$\begin{aligned} \mathsf{ind}_{\mathbf{S}^1} &: \Big(\textstyle\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\Big) \to \Big(\textstyle\prod_{(x:\mathbf{S}^1)} P(x)\Big) \\ \mathsf{comp}_{\mathbf{S}^1} &: \mathsf{dgen}_{\mathbf{S}^1} \circ \mathsf{ind}_{\mathbf{S}^1} \sim \mathsf{id} \end{aligned}$$

Therefore it remains to construct a homotopy

$$\mathsf{ind}_{\mathbf{S}^1} \circ \mathsf{dgen}_{\mathbf{S}^1} \sim \mathsf{id}.$$

Thus, for any $f : \prod_{(x:\mathbf{S}^1)} P(x)$ our task is to construct an identification

$$\mathsf{ind}_{\mathbf{S}^1}(\mathsf{dgen}_{\mathbf{S}^1}(f)) = f.$$

By function extensionality it suffices to construct a homotopy

$$\textstyle\prod_{(x:\mathbf{S}^1)} \mathsf{ind}_{\mathbf{S}^1}(\mathsf{dgen}_{\mathbf{S}^1}(f))(x) = f(x).$$

We proceed by the induction principle of the circle using the family of types $E_{g,f}(x) :\equiv g(x) = f(x)$ indexed by $x : \mathbf{S}^1$, where g is the function

$$g :\equiv \mathsf{ind}_{\mathbf{S}^1}(\mathsf{dgen}_{\mathbf{S}^1}(f)).$$

Thus, it suffices to construct

$$\begin{aligned} \alpha &: g(\mathsf{base}) = f(\mathsf{base}) \\ \beta &: \mathsf{tr}_{E_{g,f}}(\mathsf{loop}, \alpha) = \alpha. \end{aligned}$$

An argument by path induction on p yields that

$$\Big(\mathsf{apd}_g(p) \cdot r = \mathsf{ap}_{\mathsf{tr}_P(p)}(q) \cdot \mathsf{apd}_f(p)\Big) \to \Big(\mathsf{tr}_{E_{g,f}}(p, q) = r\Big),$$

for any $f, g : \prod_{(x:X)} P(x)$ and any $p : x = x'$, $q : g(x) = f(x)$ and $r : g(x') = f(x')$. Therefore it suffices to construct an identification $\alpha : g(\mathsf{base}) = f(\mathsf{base})$ equipped with an identification β witnessing that the square

$$\begin{tikzcd} \mathsf{tr}_P(\mathsf{loop}, g(\mathsf{base})) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "\mathsf{apd}_g(\mathsf{loop})"'] & \mathsf{tr}_P(\mathsf{loop}, f(\mathsf{base})) \arrow[d, equal, "\mathsf{apd}_f(\mathsf{loop})"] \\ g(\mathsf{base}) \arrow[r, equal, "\alpha"'] & f(\mathsf{base})" \end{tikzcd}$$

commutes. Notice that we get exactly such a pair (α, β) from the computation rule of the circle, by Remark 18.1.3. ☐

As a corollary we obtain the following uniqueness principle for dependent functions defined by the induction principle of the circle.

Corollary 18.2.2. *Consider a type family P over the circle, and let*

$$y : P(\mathsf{base})$$

$$p : \mathsf{tr}_P(\mathsf{loop}, y) = y.$$

Then the type of functions $f : \prod_{(x:\mathbf{S}^1)} P(x)$ *equipped with an identification*

$$\alpha : f(\mathsf{base}) = y$$

and an identification β *witnessing that the square*

$$\begin{tikzcd}
\mathsf{tr}_P(\mathsf{loop}, f(\mathsf{base})) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "\mathsf{apd}_f(\mathsf{loop})"'] & \mathsf{tr}_P(\mathsf{loop}, y) \arrow[d, equal, "p"] \\
f(\mathsf{base}) \arrow[r, equal, "\alpha"'] & y
\end{tikzcd}$$

commutes, is contractible.

Now we use the dependent universal property to derive the ordinary universal property of the circle. It would be tempting to say that it is a direct corollary, but we need to address the transport that occurs in the dependent universal property.

Theorem 18.2.3. *For each type* X*, the* ***action on generators***

$$\mathsf{gen}_{\mathbf{S}^1} : (\mathbf{S}^1 \to X) \to \sum_{(x:X)} x = x$$

given by $f \mapsto (f(\mathsf{base}), \mathsf{ap}_f(\mathsf{loop}))$ *is an equivalence.*

Proof. We prove the claim by constructing a commuting triangle

$$\begin{tikzcd}
& (\mathbf{S}^1 \to X) \arrow[dl, "\mathsf{gen}_{\mathbf{S}^1}"'] \arrow[dr, "\mathsf{dgen}_{\mathbf{S}^1}"] & \\
\left(\sum_{(x:X)} x = x\right) \arrow[rr, "\simeq"'] & & \left(\sum_{(x:X)} \mathsf{tr}_{\mathsf{const}_X}(\mathsf{loop}, x) = x\right)
\end{tikzcd}$$

in which the bottom map is an equivalence. Indeed, once we have such a triangle, we use the fact from Theorem 18.2.1 that $\mathsf{dgen}_{\mathbf{S}^1}$ is an equivalence to conclude that $\mathsf{gen}_{\mathbf{S}^1}$ is an equivalence.

To construct the bottom map, we first observe that for any constant type family const_B over a type A, any $p : a = a'$ in A, and any $b : B$, there is an identification

$$\mathsf{tr}\text{-}\mathsf{const}_B(p, b) = b.$$

This identification is easily constructed by path induction on p. Now we construct the bottom map as the induced map on total spaces of the family of maps

$$l \mapsto \mathsf{tr}\text{-}\mathsf{const}_X(\mathsf{loop}, x) \cdot l,$$

indexed by $x : X$. Since concatenating by a path is an equivalence, it follows by Theorem 9.1.3 that the induced map on total spaces is indeed an equivalence.

To show that the triangle commutes, it suffices to construct for any $f : \mathbf{S}^1 \to X$ an identification witnessing that the triangle

$$\begin{tikzcd}
\mathsf{tr}_{\mathsf{const}_X}(\mathsf{loop}, f(\mathsf{base})) \arrow[rr, equal, "\mathsf{tr}\text{-}\mathsf{const}_X(\mathsf{loop}, f(\mathsf{base}))"] \arrow[dr, equal, "\mathsf{apd}_f(\mathsf{loop})"'] & & f(\mathsf{base}) \arrow[dl, equal, "\mathsf{ap}_f(\mathsf{loop})"] \\
& f(\mathsf{base}) &
\end{tikzcd}$$

commutes. This again follows from general considerations: for any $f : A \to B$ and any $p : a = a'$ in A, the triangle

$$\begin{tikzcd} \mathsf{tr}_{\mathsf{const}_B}(p, f(a)) \arrow[rr, equal, "\mathsf{tr\text{-}const}_B(p, f(a))"] \arrow[dr, equal, "\mathsf{apd}_f(p)"'] & & f(a) \\ & f(a') \arrow[ur, equal, "\mathsf{ap}_f(p)"'] & \end{tikzcd}$$

commutes by path induction on p. $\square$

Corollary 18.2.4. *For any loop $l : x = x$ in a type X, the type of maps $f : \mathbf{S}^1 \to X$ equipped with an identification*

$$\alpha : f(\mathsf{base}) = x$$

and an identification β witnessing that the square

$$\begin{tikzcd} f(\mathsf{base}) \arrow[r, equal, "\alpha"] \arrow[d, equal, "\mathsf{ap}_f(\mathsf{loop})"'] & x \arrow[d, equal, "l"] \\ f(\mathsf{base}) \arrow[r, equal, "\alpha"'] & x \end{tikzcd}$$

commutes, is contractible.

18.3 Multiplication on the circle

One way the circle arises classically, is as the set of complex numbers at distance 1 from the origin. It is an elementary fact that $|xy| = |x||y|$ for any two complex numbers $x, y \in \mathbb{C}$, so it follows that when we multiply two complex numbers that both lie on the unit circle, then the result lies again on the unit circle. Thus, using complex multiplication we see that there is a multiplication operation on the circle. And there is a shadow of this operation in type theory, even though our circle arises in a very different way!

Definition 18.3.1. We define a binary operation

$$\mathsf{mul}_{\mathbf{S}^1} : \mathbf{S}^1 \to (\mathbf{S}^1 \to \mathbf{S}^1).$$

Construction. Using the universal property of the circle, we define $\mathsf{mul}_{\mathbf{S}^1}$ as the unique map $\mathbf{S}^1 \to (\mathbf{S}^1 \to \mathbf{S}^1)$ equipped with an identification

$$\mathsf{base\text{-}mul}_{\mathbf{S}^1} : \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}) = \mathsf{id}$$

and an identification $\mathsf{loop\text{-}mul}_{\mathbf{S}^1}$ witnessing that the square

$$\begin{tikzcd} \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}) \arrow[r, equal, "\mathsf{base\text{-}mul}_{\mathbf{S}^1}"] \arrow[d, equal, "\mathsf{ap}_{\mathsf{mul}_{\mathbf{S}^1}}(\mathsf{loop})"'] & \mathsf{id} \arrow[d, equal, "\mathsf{eq\text{-}htpy}(H)"] \\ \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}) \arrow[r, equal, "\mathsf{base\text{-}mul}_{\mathbf{S}^1}"'] & \mathsf{id} \end{tikzcd}$$

commutes. Note that in this square we have a homotopy $H : \mathsf{id} \sim \mathsf{id}$, which is not yet defined. We use the dependent universal property of the circle with respect to the family $E_{\mathsf{id},\mathsf{id}}$ given by

$$E_{\mathsf{id},\mathsf{id}}(x) :\equiv (x = x),$$

to define H as the unique homotopy equipped with an identification

$$\alpha : H(\mathsf{base}) = \mathsf{loop}$$

and an identification β witnessing that the square

$$\begin{tikzcd}
\mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop}, H(\mathsf{base})) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "\mathsf{apd}_H(\mathsf{loop})"'] & \mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop}, \mathsf{loop}) \arrow[d, equal, "\gamma"] \\
H(\mathsf{base}) \arrow[r, equal, "\alpha"'] & \mathsf{loop}
\end{tikzcd}$$

commutes. Now it remains to define the path $\gamma : \mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop}, \mathsf{loop}) = \mathsf{loop}$ in the above square. To proceed, we first observe that a simple path induction argument yields a function

$$\Big(p \cdot r = q \cdot p\Big) \to \Big(\mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(p, q) = r\Big),$$

for any $p : \mathsf{base} = x$, $q : \mathsf{base} = \mathsf{base}$ and $r : x = x$. In particular, we have a function

$$\Big(\mathsf{loop} \cdot \mathsf{loop} = \mathsf{loop} \cdot \mathsf{loop}\Big) \to \Big(\mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop}, \mathsf{loop}) = \mathsf{loop}\Big).$$

Now we apply this function to $\mathsf{refl}_{\mathsf{loop} \cdot \mathsf{loop}}$ to obtain the desired identification

$$\gamma : \mathsf{tr}_{E_{\mathsf{id},\mathsf{id}}}(\mathsf{loop}, \mathsf{loop}) = \mathsf{loop}. \qquad \square$$

Remark 18.3.2. In the definition of $H : \mathsf{id} \sim \mathsf{id}$ above, it is important that we didn't choose H to be $\mathsf{refl\text{-}htpy}$. If we had done so, the resulting operation would be homotopic to $x, y \mapsto y$, which is clearly not what we had in mind with the multiplication operation on the circle. See also Exercise 18.2.

The left unit law $\mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, x) = x$ holds by the computation rule of the universal property. More precisely, we define

$$\mathsf{left\text{-}unit}_{\mathbf{S}^1} :\equiv \mathsf{htpy\text{-}eq}(\mathsf{base\text{-}mul}_{\mathbf{S}^1}).$$

For the right unit law, however, we need to give a separate argument that is surprisingly involved, because all the aspects of the definition of $\mathsf{mul}_{\mathbf{S}^1}$ will come out and play their part.

Theorem 18.3.3. *The multiplication operation on the circle satisfies the right unit law, i.e., we have*

$$\mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{base}) = x$$

for any $x : \mathbf{S}^1$.

Proof. The proof is by induction on the circle. In the base case we use the left unit law

$$\mathsf{left\text{-}unit}_{\mathbf{S}^1}(\mathsf{base}) : \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, \mathsf{base}) = \mathsf{base}.$$

Thus, it remains to show that

$$\mathsf{tr}_P(\mathsf{loop}, \mathsf{left\text{-}unit}_{\mathbf{S}^1}(\mathsf{base})) = \mathsf{left\text{-}unit}_{\mathbf{S}^1}(\mathsf{base}),$$

where P is the family over the circle given by

$$P(x) :\equiv \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{base}) = x.$$

Now we observe that there is a function

$$\Big(\mathsf{htpy\text{-}eq}(\mathsf{ap}_{\mathsf{mul}_{\mathbf{S}^1}}(p))(\mathsf{base}) \cdot r = q \cdot p\Big) \to \Big(\mathsf{tr}_P(p,q) = r\Big),$$

for any

$$\begin{aligned} p &: \mathsf{base} = x \\ q &: \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base},\mathsf{base}) = \mathsf{base} \\ r &: \mathsf{mul}_{\mathbf{S}^1}(x,\mathsf{base}) = x. \end{aligned}$$

Thus we see that, in order to construct an identification

$$\mathsf{tr}_P(\mathsf{loop}, \mathsf{left\text{-}unit}_{\mathbf{S}^1}) = \mathsf{left\text{-}unit}_{\mathbf{S}^1},$$

it suffices to show that the square

$$\begin{tikzcd} \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base},\mathsf{base}) \arrow[r, equal, "\mathsf{left\text{-}unit}_{\mathbf{S}^1}(\mathsf{base})"] \arrow[d, equal, "\mathsf{htpy\text{-}eq}(\mathsf{ap}_{\mathsf{mul}_{\mathbf{S}^1}}(\mathsf{loop}))(\mathsf{base})"'] & \mathsf{base} \arrow[d, equal, "\mathsf{loop}"] \\ \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base},\mathsf{base}) \arrow[r, equal, "\mathsf{left\text{-}unit}_{\mathbf{S}^1}(\mathsf{base})"'] & \mathsf{base} \end{tikzcd}$$

commutes. Now we note that we have an identification $H(\mathsf{base}) = \mathsf{loop}$. It is indeed at this point, where it is important that H is not the trivial homotopy, because now we can proceed by observing that the above square commutes if and only if the square

$$\begin{tikzcd} \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base},\mathsf{base}) \arrow[r, equal, "\mathsf{htpy\text{-}eq}(\mathsf{base\text{-}mul}_{\mathbf{S}^1})(\mathsf{base})"] \arrow[d, equal, "\mathsf{htpy\text{-}eq}(\mathsf{ap}_{\mathsf{mul}_{\mathbf{S}^1}}(\mathsf{loop}))(\mathsf{base})"'] & \mathsf{base} \arrow[d, equal, "H(\mathsf{base})"] \\ \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base},\mathsf{base}) \arrow[r, equal, "\mathsf{htpy\text{-}eq}(\mathsf{base\text{-}mul}_{\mathbf{S}^1})(\mathsf{base})"'] & \mathsf{base} \end{tikzcd}$$

commutes. The commutativity of this square easily follows from the identification $\mathsf{loop\text{-}mul}_{\mathbf{S}^1}$ constructed in Definition 18.3.1. □

Exercises

18.1 (a) Let $P : \mathbf{S}^1 \to \mathsf{Prop}$ be a family of propositions over the circle. Show that

$$P(\mathsf{base}) \to \prod_{(x:\mathbf{S}^1)} P(x).$$

In this sense the circle is *connected*.

(b) Show that any embedding $m : \mathbf{S}^1 \to \mathbf{S}^1$ is an equivalence.

(c) Show that for any embedding $m : X \to \mathbf{S}^1$, there is a proposition P and an equivalence $e : X \simeq \mathbf{S}^1 \times P$ for which the triangle

$$\begin{tikzcd} X \arrow[rr, "e"] \arrow[dr, "m"'] & & \mathbf{S}^1 \times P \arrow[dl, "\mathsf{pr}_1"] \\ & \mathbf{S}^1 & \end{tikzcd}$$

commutes. In other words, all the embeddings into the circle are of the form $\mathbf{S}^1 \times P \to \mathbf{S}^1$.

18.2 Show that for any type X and any $x : X$, the map

$$\mathsf{ind}_{\mathbf{S}^1}(x, \mathsf{refl}_x) : \mathbf{S}^1 \to X$$

is homotopic to the constant map const_x.

18.3 (a) Show that for any $x : \mathbf{S}^1$, both functions

$$\mathsf{mul}_{\mathbf{S}^1}(x, -) \qquad \text{and} \qquad \mathsf{mul}_{\mathbf{S}^1}(-, x)$$

are equivalences.

(b) Show that the function

$$\mathsf{mul}_{\mathbf{S}^1} : \mathbf{S}^1 \to (\mathbf{S}^1 \to \mathbf{S}^1)$$

is an embedding. Compare this fact with Exercise 17.2.

(c) Show that multiplication on the circle is associative and commutative.

18.4 (a) Show that a type X is a set if and only if the map

$$\lambda x.\, \lambda t.\, x : X \to (\mathbf{S}^1 \to X)$$

is an equivalence.

(b) Show that a type X is a set if and only if the map

$$\lambda f.\, f(\mathsf{base}) : (\mathbf{S}^1 \to X) \to X$$

is an equivalence.

18.5 Show that the multiplicative operation on the circle is commutative, i.e. construct an identification

$$\mathsf{mul}_{\mathbf{S}^1}(x, y) = \mathsf{mul}_{\mathbf{S}^1}(y, x).$$

for every $x, y : \mathbf{S}^1$.

18.6 Show that the circle, equipped with the multiplicative operation $\mathsf{mul}_{\mathbf{S}^1}$ is an abelian group, i.e. construct an inverse operation

$$\mathsf{inv}_{\mathbf{S}^1} : \mathbf{S}^1 \to \mathbf{S}^1$$

and construct identifications

$$\mathsf{left\text{-}inv}_{\mathbf{S}^1} : \mathsf{mul}_{\mathbf{S}^1}(\mathsf{inv}_{\mathbf{S}^1}(x), x) = \mathsf{base}$$

$$\mathsf{right\text{-}inv}_{\mathbf{S}^1} : \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{inv}_{\mathbf{S}^1}(x)) = \mathsf{base}.$$

Moreover, show that the square

$$\begin{tikzcd}
\mathsf{inv}_{\mathbf{S}^1}(\mathsf{base}) \arrow[r, equal] \arrow[d, equal] & \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, \mathsf{inv}_{\mathbf{S}^1}(\mathsf{base})) \arrow[d, equal] \\
\mathsf{mul}_{\mathbf{S}^1}(\mathsf{inv}_{\mathbf{S}^1}(\mathsf{base}), \mathsf{base}) \arrow[r, equal] & \mathsf{base}
\end{tikzcd}$$

commutes.

18.7 Show that for any multiplicative operation

$$\mu : \mathbf{S}^1 \to (\mathbf{S}^1 \to \mathbf{S}^1)$$

that satisfies the condition that $\mu(x, -)$ and $\mu(-, x)$ are equivalences for any $x : \mathbf{S}^1$, there is a term $e : \mathbf{S}^1$ such that

$$\mu(x, y) = \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(\bar{e}, y))$$

for every $x, y : \mathbf{S}^1$, where $\bar{e} :\equiv \mathsf{inv}_{\mathbf{S}^1}(e)$ is the complex conjucation of e on $\mathbf{S}^1$.

19 The fundamental cover of the circle

In this section we show that the loop space of the circle is equivalent to $\mathbb{Z}$ by constructing the universal cover of the circle as an application of the univalence axiom.

19.1 Families over the circle

The type of small families over $\mathbf{S}^1$ is just the function type $\mathbf{S}^1 \to \mathcal{U}$, so in fact we may use the universal property of the circle to construct small dependent types over the circle. By the universal property, small type families over $\mathbf{S}^1$ are equivalently described as pairs (X, p) consisting of a type $X : \mathcal{U}$ and an identification $p : X = X$. This is where the univalence axiom comes in. By the map

$$\mathsf{eq\text{-}equiv}_{X,X} : (X \simeq X) \to (X = X)$$

it suffices to provide an equivalence $X \simeq X$.

Definition 19.1.1. Consider a type X and every equivalence $e : X \simeq X$. We will construct a dependent type $\mathcal{D}(X,e) : \mathbf{S}^1 \to \mathcal{U}$ with an equivalence $x \mapsto x_{\mathcal{D}} : X \simeq \mathcal{D}(X, e, \mathsf{base})$ for which the square

$$\begin{tikzcd} X \arrow[r, "\simeq"] \arrow[d, "e"'] & \mathcal{D}(X,e,\mathsf{base}) \arrow[d, "\mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop})"] \\ X \arrow[r, "\simeq"'] & \mathcal{D}(X,e,\mathsf{base}) \end{tikzcd}$$

commutes. We also write $d \mapsto d_X$ for the inverse of this equivalence, so that the relations

$$(x_{\mathcal{D}})_X = x \qquad\qquad (e(x)_{\mathcal{D}}) = \mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop}, x_{\mathcal{D}})$$
$$(d_X)_{\mathcal{D}} = d \qquad\qquad (\mathsf{tr}_{\mathcal{D}(X,e)}(d))_X = e(d_X)$$

hold.

The type $\sum_{(X:\mathcal{U})} X \simeq X$ is also called the type of **descent data** for the circle.

Construction. An easy path induction argument reveals that

$$\mathsf{equiv\text{-}eq}(\mathsf{ap}_P(\mathsf{loop})) = \mathsf{tr}_P(\mathsf{loop})$$

for each dependent type $P : \mathbf{S}^1 \to \mathcal{U}$. Therefore we see that the triangle

$$\begin{tikzcd} & (\mathbf{S}^1 \to \mathcal{U}) \arrow[dl, "\mathsf{gen}_{\mathbf{S}^1}"'] \arrow[dr, "\mathsf{desc}_{\mathbf{S}^1}"] & \\ \sum_{(X:\mathcal{U})} X = X \arrow[rr, "\mathsf{tot}(\lambda X.\, \mathsf{equiv\text{-}eq}_{X,X})"'] & & \sum_{(X:\mathcal{U})} X \simeq X \end{tikzcd}$$

commutes, where the map $\mathsf{desc}_{\mathbf{S}^1}$ is given by $P \mapsto (P(\mathsf{base}), \mathsf{tr}_P(\mathsf{loop}))$ and the bottom map is an equivalence by the univalence axiom and Theorem 9.1.3. Now it follows by the 3-for-2 property that $\mathsf{desc}_{\mathbf{S}^1}$ is an equivalence, since $\mathsf{gen}_{\mathbf{S}^1}$ is an equivalence by Theorem 18.2.3. This means that for every type X and every $e : X \simeq X$ there is a type family $\mathcal{D}(X,e) : \mathbf{S}^1 \to \mathcal{U}$ such that

$$(\mathcal{D}(X, e, \mathsf{base}), \mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop})) = (X, e).$$

Equivalently, we have $p : \mathcal{D}(X, e, \mathsf{base}) = X$ and $\mathsf{tr}(p, \mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop})) = e$. Thus, we obtain $\mathsf{equiv\text{-}eq}(p) : \mathcal{D}(X, e, \mathsf{base}) \simeq X$, for which the square

$$\begin{tikzcd}
\mathcal{D}(X, e, \mathsf{base}) \arrow[r, "\mathsf{equiv\text{-}eq}(p)"] \arrow[d, "\mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop})"'] & X \arrow[d, "e"] \\
\mathcal{D}(X, e, \mathsf{base}) \arrow[r, "\mathsf{equiv\text{-}eq}(p)"'] & X
\end{tikzcd}$$

commutes. □

19.2 The fundamental cover of the circle

The *fundamental cover* of the circle is a family of sets over the circle with contractible total space. Classically, the fundamental cover is described as a map $\mathbb{R} \to \mathbf{S}^1$ that winds the real line around the circle. In homotopy type theory there is no analogue of such a construction.

Recall from Exercise 7.6 that the successor function $\mathsf{succ} : \mathbb{Z} \to \mathbb{Z}$ is an equivalence. Its inverse is the predecessor function defined in Exercise 4.4.

Definition 19.2.1. The **fundamental cover** of the circle is the dependent type $\mathcal{E}_{\mathbf{S}^1} :\equiv \mathcal{D}(\mathbb{Z}, \mathsf{succ}) : \mathbf{S}^1 \to \mathcal{U}$.

Remark 19.2.2. The fundamental cover of the circle comes equipped with an equivalence

$$e : \mathbb{Z} \simeq \mathcal{E}_{\mathbf{S}^1}(\mathsf{base})$$

and a homotopy witnessing that the square

$$\begin{tikzcd}
\mathbb{Z} \arrow[r, "e"] \arrow[d, "\mathsf{succ}"'] & \mathcal{E}_{\mathbf{S}^1}(\mathsf{base}) \arrow[d, "\mathsf{tr}_{\mathcal{E}_{\mathbf{S}^1}}(\mathsf{loop})"] \\
\mathbb{Z} \arrow[r, "e"'] & \mathcal{E}_{\mathbf{S}^1}(\mathsf{base})
\end{tikzcd}$$

commutes.

For convenience, we write $k_{\mathcal{E}}$ for the term $e(k) : \mathcal{E}_{\mathbf{S}^1}(\mathsf{base})$, for any $k : \mathbb{Z}$.

The picture of the fundamental cover is that of a helix over the circle. This picture emerges from the path liftings of loop in the total space. The segments of the helix connecting k to $k+1$ in the total space of the helix, are constructed in the following lemma.

Lemma 19.2.3. *For any $k : \mathbb{Z}$, there is an identification*

$$\mathsf{segment\text{-}helix}_k : (\mathsf{base}, k_{\mathcal{E}}) = (\mathsf{base}, \mathsf{succ}(k)_{\mathcal{E}})$$

in the total space $\sum_{(t:\mathbf{S}^1)} \mathcal{E}(t)$.

Proof. By Theorem 7.3.4 it suffices to show that

$$\prod_{(k:\mathbb{Z})} \sum_{(\alpha:\mathsf{base}=\mathsf{base})} \mathsf{tr}_{\mathcal{E}}(\alpha, k_{\mathcal{E}}) = \mathsf{succ}(k)_{\mathcal{E}}.$$

We just take $\alpha :\equiv \mathsf{loop}$. Then we have $\mathsf{tr}_{\mathcal{E}}(\alpha, k_{\mathcal{E}}) = \mathsf{succ}(k)_{\mathcal{E}}$ by the commuting square provided in the definition of $\mathcal{E}$. □

19.3 Contractibility of general total spaces

Consider a type X, a family P over X, and a term $c : \sum_{(x:X)} P(x)$, and suppose our goal is to construct a contraction

$$\prod_{(t:\sum_{(x:X)} P(x))} c = t.$$

Of course, the first step is to apply the induction principle of Σ-types, so it suffices to construct a term of type

$$\prod_{(x:X)} \prod_{(y:P(x))} c = (x, y).$$

In the case where P is the fundamental cover of the circle, we are given an equivalence $e : \mathbb{Z} \simeq \mathcal{E}(\mathsf{base})$. Using this equivalence, we obtain an equivalence

$$\Big(\prod_{(y:\mathcal{E}(y))} c = (\mathsf{base}, y)\Big) \to \Big(\prod_{(k:\mathbb{Z})} c = (\mathsf{base}, k_{\mathcal{E}})\Big).$$

More generally, if we are given an equivalence $e : F \simeq P(x)$ for some $x : X$, then we have an equivalence

$$\Big(\prod_{(y:P(x))} c = (x, y)\Big) \to \Big(\prod_{(y:F)} c = (x, e(y))\Big) \tag{19.1}$$

by precomposing with the equivalence e. Therefore we can construct a term of type $\prod_{(y:P(x))} c = (x, y)$ by constructing a term of type $\prod_{(y:F)} c = (x, e(y))$.

Furthermore, if we consider a path $p : x = x'$ in X and a commuting square

$$\begin{tikzcd} F \arrow[r, "e"] \arrow[d, "f"'] & P(x) \arrow[d, "\mathsf{tr}_P(p)"] \\ F' \arrow[r, "e'"] & P(x') \end{tikzcd}$$

where e, e', and f are all equivalences, then we obtain a function

$$\psi : \Big(\prod_{(y:F)} c = (x, e(y))\Big) \to \Big(\prod_{(y':F')} c = (x, e'(y'))\Big).$$

The function ψ is constructed as follows. Given $h : \prod_{(y:F)} c = (x, e(y))$ and $y' : F'$ we have the path $h(f^{-1}(y')) : c = (x, e(f^{-1}(y')))$. Moreover, writing G for the homotopy $f \circ f^{-1} \sim \mathsf{id}$, we have the path

$$\mathsf{tr}_P(p, e(f^{-1}(y'))) \xlongequal{H(f^{-1}(y'))} e'(f(f^{-1}(y'))) \xlongequal{\mathsf{ap}_{e'}(G(y'))} e'(y').$$

From this concatenated path we obtain the path

$$(x, e(f^{-1}(y'))) \xlongequal{\mathsf{eq\text{-}pair}(p, H(f^{-1}(y')) \cdot \mathsf{ap}_{e'}(G(y')))} (x', e'(y')).$$

Now we define the function ψ by

$$h \mapsto \lambda y'.\, h(f^{-1}(y')) \cdot \mathsf{eq\text{-}pair}(p, H(f^{-1}(y')) \cdot \mathsf{ap}_{e'}(G(y'))).$$

Note that ψ is an equivalence, since it is given as precomposition by the equivalence f^{-1}, followed by postcomposition by concatenation, which is also an equivalence. Now we state the main technical result of this section, which will help us prove the contractibility of the total space of the fundamental cover of the circle by computing transport in the family $x \mapsto \prod_{(y:P(x))} c = (x, y)$.

Definition 19.3.1. Consider a path $p : x = x'$ in X and a commuting square

$$\begin{tikzcd} F \arrow[r, "e"] \arrow[d, "f"'] & P(x) \arrow[d, "\mathsf{tr}_P(p)"] \\ F' \arrow[r, "e'"] & P(x') \end{tikzcd}$$

with $H : e' \circ f \ \mathsf{tr}_P(p) \circ e$, where e, e', and f are all equivalences. Then there is for any $y : F$ an identification

$$\mathsf{segment\text{-}tot}(y) : (x, e(y)) = (x', e'(f(y)))$$

defined as $\mathsf{segment\text{-}tot}(y) :\equiv \mathsf{eq\text{-}pair}(p, H(y)^{-1})$.

Lemma 19.3.2. *Consider a path $p : x = x'$ in X and a commuting square*

$$\begin{tikzcd} F \arrow[r, "e"] \arrow[d, "f"'] & P(x) \arrow[d, "\mathsf{tr}_P(p)"] \\ F' \arrow[r, "e'"] & P(x') \end{tikzcd}$$

with $H : e' \circ f \ \mathsf{tr}_P(p) \circ e$, where e, e', and f are all equivalences. Furthermore, let

$$\begin{aligned} h &: \textstyle\prod_{(y:F)} c = (x, e(y)) \\ h' &: \textstyle\prod_{(y':F')} c = (x', e'(y')). \end{aligned}$$

Then there is an equivalence

$$\Big(\textstyle\prod_{(y:F)} h'(f(y)) = h(y) \cdot \mathsf{segment\text{-}tot}(y)\Big) \simeq \Big(\mathsf{tr}_C(p, \varphi(h)) = \varphi'(h')\Big).$$

Proof. We first note that we have a commuting square

$$\begin{tikzcd} \prod_{(y:B(x))} c = (x, y) \arrow[r, "- \circ e"] \arrow[d, "\mathsf{tr}_C(p)"'] & \prod_{(y:F)} c = (x, e(y)) \\ \prod_{(y':B(x'))} c = (x', y') \arrow[r, "- \circ e'"'] & \prod_{(y':F')} c = (x', e'(y')) \arrow[u, "\psi"'] \end{tikzcd}$$

where $\psi(h') = \lambda y.\, h'(f(y)) \cdot \mathsf{segment\text{-}tot}(y)^{-1}$. All the maps in this square are equivalences. In particular, the inverses of the top and bottom maps are φ and φ', respectively. The claim follows from this observation, but we will spell out the details.

Since any equivalence is an embedding, we see immediately that the type $\mathsf{tr}_C(p)(\varphi(h)) = \varphi'(h')$ is equivalent to the type

$$\psi(\mathsf{tr}_C(p)(\varphi(h)) \circ e') = \psi(\varphi'(h') \circ e').$$

By the commutativity of the square, the left hand side is h. The right hand side is $\psi(h')$. Therefore it follows that

$$\begin{aligned} \Big(\mathsf{tr}_C(p)(\varphi(h)) = \varphi'(h')\Big) &\simeq \Big(h = \lambda y.\, h'(f(y)) \cdot \mathsf{segment\text{-}tot}(y)^{-1}\Big) \\ &\simeq \Big(h' \circ f \sim (\lambda y.\, h(y) \cdot \mathsf{segment\text{-}tot}(y)\Big). \end{aligned}$$ $\square$

Applying these observations to the fundamental cover of the circle, we obtain the following lemma that we will use to prove that the total space of $\mathcal{E}$ is contractible.

Corollary 19.3.3. *In order to show that the total space of $\mathcal{E}$ is contractible, it suffices to construct a function*

$$h : \prod_{(k:\mathbb{Z})} (\mathsf{base}, 0_{\mathcal{E}}) = (\mathsf{base}, k_{\mathcal{E}})$$

equipped with a homotopy

$$H : \prod_{(k:\mathbb{Z})} h(\mathsf{succ}(k)_{\mathcal{E}}) = h(k) \cdot \mathsf{segment\text{-}helix}(k).$$

In the next section we establish the dependent universal property of the integers, which we will use with Corollary 19.3.3 to show that the total space of the fundamental cover is contractible.

19.4 The dependent universal property of the integers

Lemma 19.4.1. *Let B be a family over $\mathbb{Z}$, equipped with a term $b_0 : B(0)$, and an equivalence*

$$e_k : B(k) \simeq B(\mathsf{succ}(k))$$

for each $k : \mathbb{Z}$. Then there is a dependent function $f : \prod_{(k:\mathbb{Z})} B(k)$ equipped with identifications $f(0) = b_0$ and

$$f(\mathsf{succ}(k)) = e_k(f(k))$$

for any $k : \mathbb{Z}$.

Proof. The map is defined using the induction principle for the integers, stated in Lemma 4.5.3. First we take

$$\begin{aligned} f(-1) &:\equiv e^{-1}(b_0) \\ f(0) &:\equiv b_0 \\ f(1) &:\equiv e(b_0). \end{aligned}$$

For the induction step on the negative integers we use

$$\lambda n.\, e^{-1}_{\mathsf{neg}(S(n))} : \prod_{(n:\mathbb{N})} B(\mathsf{neg}(n)) \to B(\mathsf{neg}(S(n)))$$

For the induction step on the positive integers we use

$$\lambda n.\, e(\mathsf{pos}(n)) : \prod_{(n:\mathbb{N})} B(\mathsf{pos}(n)) \to B(\mathsf{pos}(S(n))).$$

The computation rules follow in a straightforward way from the computation rules of $\mathbb{Z}$-induction and the fact that e^{-1} is an inverse of e. □

Example 19.4.2. For any type A, we obtain a map $f : \mathbb{Z} \to A$ from any $x : A$ and any equivalence $e : A \simeq A$, such that $f(0) = x$ and the square

$$\begin{tikzcd} \mathbb{Z} \arrow[r, "f"] \arrow[d, "\mathsf{succ}"'] & A \arrow[d, "e"] \\ \mathbb{Z} \arrow[r, "f"'] & A \end{tikzcd}$$

commutes. In particular, if we take $A \equiv (x = x)$ for some $x : X$, then for any $p : x = x$ we have the equivalence $\lambda q.\, p \cdot q : (x = x) \to (x = x)$. This equivalence induces a map

$$k \mapsto p^k : \mathbb{Z} \to (x = x),$$

for any $p : x = x$. This induces the **degree k map** on the circle

$$\mathsf{deg}(k) : \mathbf{S}^1 \to \mathbf{S}^1,$$

for any $k : \mathbb{Z}$, see **??**.

In the following theorem we show that the dependent function constructed in Lemma 19.4.1 is unique.

Theorem 19.4.3. *Consider a type family $B : \mathbb{Z} \to \mathcal{U}$ equipped with $b : B(0)$ and a family of equivalences*

$$e : \prod_{(k:\mathbb{Z})} B(k) \simeq B(\mathsf{succ}(k)).$$

Then the type

$$\sum_{(f:\prod_{(k:\mathbb{Z})} B(k))} (f(0) = b) \times \prod_{(k:\mathbb{Z})} f(\mathsf{succ}(k)) = e_k(f(k))$$

is contractible.

Proof. In Lemma 19.4.1 we have already constructed a term of the asserted type. Therefore it suffices to show that any two terms of this type can be identified. Note that the type $(f, p, H) = (f', p', H')$ is equivalent to the type

$$\sum_{(K:f\sim f')} (K(0) = p \cdot (p')^{-1}) \times \prod_{(k:\mathbb{Z})} K(\mathsf{succ}(k)) = (H(k) \cdot \mathsf{ap}_{e_k}(K(k))) \cdot H'(k)^{-1}.$$

We obtain a term of this type by applying Lemma 19.4.1 to the family C over $\mathbb{Z}$ given by $C(k) :\equiv f(k) = f'(k)$, which comes equipped with a base point

$$p \cdot (p')^{-1} : C(0),$$

and the family of equivalences

$$\lambda(\alpha : f(k) = f'(k)).\, (H(k) \cdot \mathsf{ap}_{e_k}(\alpha)) \cdot H'(k)^{-1} : \prod_{(k:\mathbb{Z})} C(k) \simeq C(\mathsf{succ}(k)).$$ □

One way of phrasing the following corollary, is that $\mathbb{Z}$ is the 'initial type equipped with a point and an automorphism'.

Corollary 19.4.4. *For any type X equipped with a base point $x_0 : X$ and an automorphism $e : X \simeq X$, the type*

$$\sum_{(f:\mathbb{Z}\to X)} (f(0) = x_0) \times ((f \circ \mathsf{succ}) \sim (e \circ f))$$

is contractible.

19.5 The identity type of the circle

Lemma 19.5.1. *The total space $\sum_{(t:\mathbf{S}^1)} \mathcal{E}(t)$ of the fundamental cover of $\mathbf{S}^1$ is contractible.*

Proof. By Corollary 19.3.3 it suffices to construct a function

$$h : \prod_{(k:\mathbb{Z})} (\mathsf{base}, 0_{\mathcal{E}}) = (\mathsf{base}, k_{\mathcal{E}})$$

equipped with a homotopy

$$H : \prod_{(k:\mathbb{Z})} h(\mathsf{succ}(k)_{\mathcal{E}}) = h(k) \cdot \mathsf{segment\text{-}helix}(k).$$

We obtain h and H by the elimination principle of Lemma 19.4.1. Indeed, the family P over the integers given by $P(k) :\equiv (\mathsf{base}, 0_{\mathcal{E}}) = (\mathsf{base}, k_{\mathcal{E}})$ comes equipped with a term $\mathsf{refl}_{(\mathsf{base}, 0_{\mathcal{E}})} : P(0)$, and a family of equivalences

$$\prod_{(k:\mathbb{Z})} P(k) \simeq P(\mathsf{succ}(k))$$

given by $k, p \mapsto p \cdot \mathsf{segment\text{-}helix}(k)$. □

Theorem 19.5.2. *The family of maps*

$$\prod_{(t:\mathbf{S}^1)} (\mathsf{base} = t) \to \mathcal{E}(t)$$

sending $\mathsf{refl}_{\mathsf{base}}$ *to* $0_{\mathcal{E}}$ *is a family of equivalences. In particular, the loop space of the circle is equivalent to* $\mathbb{Z}$.

Proof. This is a direct corollary of Lemma 19.5.1 and Theorem 9.2.2. □

Corollary 19.5.3. *The circle is a* 1*-type and not a* 0*-type.*

Proof. To see that the circle is a 1-type we have to show that $s = t$ is a 0-type for every $s, t : \mathbf{S}^1$. By Exercise 18.1 it suffices to show that the loop space of the circle is a 0-type. This is indeed the case, because $\mathbb{Z}$ is a 0-type, and we have an equivalence $(\mathsf{base} = \mathsf{base}) \simeq \mathbb{Z}$.

Furthermore, since $\mathbb{Z}$ is a 0-type and not a (-1)-type, it follows that the circle is a 1-type and not a 0-type. □

Exercises

19.1 Show that the map

$$\mathbb{Z} \to \Omega(\mathbf{S}^1)$$

is a group homomorphism. Conclude that the loop space $\Omega(\mathbf{S}^1)$ as a group is isomorphic to $\mathbb{Z}$.

19.2 (a) Show that

$$\prod_{(x:\mathbf{S}^1)} \neg\neg(\mathsf{base} = x).$$

(b) On the other hand, use the fundamental cover of the circle to show that

$$\neg\Big(\prod_{(x:\mathbf{S}^1)} \mathsf{base} = x\Big).$$

(c) Conclude that

$$\neg\Big(\prod_{(X:\mathcal{U})} \neg\neg X \to X\Big)$$

for any univalent universe $\mathcal{U}$ containing the circle.

19.3 (a) Show that for every $x : X$, we have an equivalence

$$\Big(\sum_{(f:\mathbf{S}^1 \to X)} f(\mathsf{base}) = x\Big) \simeq (x = x)$$

(b) Show that for every $t : \mathbf{S}^1$, we have an equivalence

$$\Big(\textstyle\sum_{(f:\mathbf{S}^1\to\mathbf{S}^1)} f(\mathsf{base}) = t\Big) \simeq \mathbb{Z}$$

The base point preserving map $f : \mathbf{S}^1 \to \mathbf{S}^1$ corresponding to $k : \mathbb{Z}$ is called the **degree** k **map** on the circle, and is denoted by $\mathsf{deg}(k)$.

(c) Show that for every $t : \mathbf{S}^1$, we have an equivalence

$$\Big(\textstyle\sum_{(e:\mathbf{S}^1\simeq\mathbf{S}^1)} e(\mathsf{base}) = t\Big) \simeq \mathbf{2}$$

19.4 The **(twisted) double cover** of the circle is defined as the type family $\mathcal{T} :\equiv \mathcal{D}(\mathbf{2}, \mathsf{neg}) : \mathbf{S}^1 \to \mathcal{U}$, where $\mathsf{neg} : \mathbf{2} \simeq \mathbf{2}$ is the negation equivalence of Exercise 7.5.

(a) Show that $\neg(\prod_{(t:\mathbf{S}^1)} \mathcal{T}(t))$.

(b) Construct an equivalence $e : \mathbf{S}^1 \simeq \sum_{(t:\mathbf{S}^1)} \mathcal{T}(t)$ for which the triangle

$$\begin{tikzcd} \mathbf{S}^1 \arrow[rr, "e"] \arrow[dr, "\mathsf{deg}(2)"'] & & \sum_{(t:\mathbf{S}^1)} \mathcal{T}(t) \arrow[dl, "\mathsf{pr}_1"] \\ & \mathbf{S}^1 & \end{tikzcd}$$

commutes.

19.5 Construct an equivalence $(\mathbf{S}^1 \simeq \mathbf{S}^1) \simeq \mathbf{S}^1 + \mathbf{S}^1$ for which the triangle

$$\begin{tikzcd} (\mathbf{S}^1 \simeq \mathbf{S}^1) \arrow[rr, "\simeq"] \arrow[dr, "\mathsf{ev\text{-}base}"'] & & (\mathbf{S}^1 + \mathbf{S}^1) \arrow[dl, "\mathsf{fold}"] \\ & \mathbf{S}^1 & \end{tikzcd}$$

commutes. Conclude that a univalent universe containing a circle is not a 1-type.

19.6 (a) Construct a family of equivalences

$$\textstyle\prod_{(t:\mathbf{S}^1)} \big((t = t) \simeq \mathbb{Z}\big).$$

(b) Use Exercise 18.4 to show that $(\mathsf{id}_{\mathbf{S}^1} \sim \mathsf{id}_{\mathbf{S}^1}) \simeq \mathbb{Z}$.

(c) Use Exercise 11.7 to show that

$$\mathsf{has\text{-}inverse}(\mathsf{id}_{\mathbf{S}^1}) \simeq \mathbb{Z},$$

and conclude that $\mathsf{has\text{-}inverse}(\mathsf{id}_{\mathbf{S}^1}) \not\simeq \mathsf{is\text{-}equiv}(\mathsf{id}_{\mathbf{S}^1})$.

19.7 Consider a map $i : A \to \mathbf{S}^1$, and assume that i has a retraction. Construct a term of type

$$\mathsf{is\text{-}contr}(A) + \mathsf{is\text{-}equiv}(i).$$

19.8 (a) Show that the multiplicative operation on the circle is associative, i.e. construct an identification

$$\mathsf{assoc}_{\mathbf{S}^1}(x, y, z) : \mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, y), z) = \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(y, z))$$

for any $x, y, z : \mathbf{S}^1$.

(b) Show that the associator satisfies unit laws, in the sense that the following triangles commute:

$$\begin{tikzcd}
\mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, x), y) \arrow[rr, equal] \arrow[dr, equal] & & \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, \mathsf{mul}_{\mathbf{S}^1}(x, y)) \arrow[dl, equal] \\
& \mathsf{mul}_{\mathbf{S}^1}(x, y) &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{base}), y) \arrow[rr, equal] \arrow[dr, equal] & & \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(\mathsf{base}, y)) \arrow[dl, equal] \\
& \mathsf{mul}_{\mathbf{S}^1}(x, y) &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, y), \mathsf{base}) \arrow[rr, equal] \arrow[dr, equal] & & \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(y, \mathsf{base})) \arrow[dl, equal] \\
& \mathsf{mul}_{\mathbf{S}^1}(x, y). &
\end{tikzcd}$$

(c) State the laws that compute

$$\begin{gathered}
\mathsf{assoc}_{\mathbf{S}^1}(\mathsf{base}, \mathsf{base}, x) \\
\mathsf{assoc}_{\mathbf{S}^1}(\mathsf{base}, x, \mathsf{base}) \\
\mathsf{assoc}_{\mathbf{S}^1}(x, \mathsf{base}, \mathsf{base}) \\
\mathsf{assoc}_{\mathbf{S}^1}(\mathsf{base}, \mathsf{base}, \mathsf{base}).
\end{gathered}$$

Note: the first three laws should be 3-cells and the last law should be a 4-cell. The laws are automatically satisfied, since the circle is a 1-type.

19.9 Construct the **Mac Lane pentagon** for the circle, i.e. show that the pentagon

$$\begin{tikzcd}[column sep=small]
& \mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, y), z), w) \arrow[r, equal] \arrow[dl, equal] & \mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, y), \mathsf{mul}_{\mathbf{S}^1}(z, w)) \arrow[dr, equal] & \\
\mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(y, z)), w) \arrow[drr, equal] & & & \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(y, \mathsf{mul}_{\mathbf{S}^1}(z, w))) \arrow[dl, equal] \\
& & \mathsf{mul}_{\mathbf{S}^1}(x, \mathsf{mul}_{\mathbf{S}^1}(\mathsf{mul}_{\mathbf{S}^1}(y, z), w)) &
\end{tikzcd}$$

commutes for every $x, y, z, w : \mathbf{S}^1$.

19.10 Recall from Exercise 14.4 that if $f : A \to B$ is a surjective map, then the precomposition map

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

is an embedding for every set C. Give an example of a surjective map $f : A \to B$, such that the precomposition function

$$- \circ f : (B \to \mathbf{S}^1) \to (A \to \mathbf{S}^1)$$

is *not* an embedding, showing that the condition that C is a set is essential.

20 The classifying type of a group

20.1 The classifying type of a group

Theorem 20.1.1. *For every group G there is a pointed connected* 1*-type BG equipped with group isomorphism*

$$\Omega(BG) \simeq G$$

Chapter IV

Concepts of higher category theory in type theory

21 Homotopy pullbacks

Suppose we are given a map $f : A \to B$, and type families P over A, and Q over B. Then any family of maps

$$g : \prod_{(x:A)} P(x) \to Q(f(x))$$

gives rise to a commuting square

$$\begin{tikzcd} \sum_{(x:A)} P(x) \arrow[r, "\mathsf{tot}_f(g)"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(y:B)} Q(y) \arrow[d, "\mathsf{pr}_1"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

where $\mathsf{tot}_f(g)$ is defined as $\lambda(x,y).\,(f(x), g(x,y))$. In the main theorem of this chapter we show that g is a family of equivalences if and only if this square satisfies a certain universal property: the universal property of *pullback squares*.

Pullback squares are of interest because they appear in many situations. Cartesian products, fibers of maps, and substitutions can all be presented as pullbacks. Moreover, the fact that a family of maps $g : \prod_{(x:A)} P(x) \to Q(f(x))$ is a family of equivalences if and only if it induces a pullback square has the very useful corollary that a square of the form

$$\begin{tikzcd} C \arrow[r] \arrow[d, "p"'] & D \arrow[d, "q"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

is a pullback square if and only if the induced family of maps between the fibers

$$\prod_{(x:A)} \mathsf{fib}_p(x) \to \mathsf{fib}_q(f(x))$$

is a family of equivalences. This connection between pullbacks and *fiberwise equivalences* has an important role in the descent theorem in §24.

A second reason for studying pullback squares is that the dual notion of *pushouts* is an important tool to construct new types, including the n-spheres for arbitrary n. The duality of pullbacks and pushouts makes it possible to obtain proofs of many statements about pushouts from their dual statements about pullbacks.

21.1 The universal property of pullbacks

Definition 21.1.1. A **cospan** consists of three types A, X, and B, and maps $f : A \to X$ and $g : B \to X$.

Definition 21.1.2. Consider a cospan

$$A \xrightarrow{f} X \xleftarrow{g} B$$

and a type C. A **cone** on the cospan $A \to X \leftarrow B$ with **vertex** C consists of maps $p : C \to A$, $q : C \to B$ and a homotopy $H : f \circ p \sim g \circ q$ witnessing that the square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

commutes. We write

$$\mathsf{cone}(C) :\equiv \textstyle\sum_{(p:C\to A)}\sum_{(q:C\to B)} f \circ p \sim g \circ q$$

for the type of cones with vertex C.

It is good practice to characterize the identity type of any type of importance. In the following lemma we give a characterization of the identity type of the type $\mathsf{cone}(C)$ of cones on $A \to X \leftarrow B$ with vertex C. Such characterizations are entirely routine in homotopy type theory.

Lemma 21.1.3. *Let (p, q, H) and (p', q', H') be cones on a cospan $f : A \to X \leftarrow B : g$, both with vertex C. Then the type $(p, q, H) = (p', q', H')$ is equivalent to the type of triples (K, L, M) consisting of*

$$K : p \sim p'$$
$$L : q \sim q'$$

and a homotopy $M : H \cdot (g \cdot L) \sim (f \cdot K) \cdot H'$ witnessing that the square

$$\begin{tikzcd} f \circ p \arrow[r, "f \cdot K"] \arrow[d, "H"'] & f \circ p' \arrow[d, "H'"] \\ g \circ q \arrow[r, "g \cdot L"'] & g \circ q' \end{tikzcd}$$

of homotopies commutes.

Proof. By the fundamental theorem of identity types (Theorem 9.2.2) it suffices to show that the type

$$\textstyle\sum_{((p',q',H'):\sum_{(p':C\to A)}\sum_{(q':C\to B)} f\circ p' \sim g \circ q')}\sum_{(K:p\sim p')}\sum_{(L:q\sim q')} H \cdot (g \cdot L) \sim (f \cdot K) \cdot H'$$

is contractible. Using associativity of Σ-types and commutativity of cartesian products, it is easy to show that this type is equivalent to the type

$$\textstyle\sum_{((p',K):\sum_{(p':C\to A)} p\sim p')}\sum_{((q',L):\sum_{(q':C\to B)} q\sim q')}\sum_{(H:f\circ p'\sim g\circ q')} H \cdot (g \cdot L) \sim (f \cdot K) \cdot H'$$

Now we observe that the types $\sum_{(p':C\to A)} p \sim p'$ and $\sum_{(q':C\to B)} q \sim q'$ are contractible, with centers of contraction

$$(p, \mathsf{refl\text{-}htpy}_p) : \textstyle\sum_{(p':C'\to A)} p \sim p'$$
$$(q, \mathsf{refl\text{-}htpy}_q) : \textstyle\sum_{(q':C'\to B)} q \sim q'.$$

Thus we apply Exercise 8.5 to see that the type of tuples $((p', K), (q', L), (H', M))$ is equivalent to the type

$$\textstyle\sum_{(H':f\circ p'\sim g\circ q')} H \cdot \mathsf{refl\text{-}htpy}_{g\circ q} \sim \mathsf{refl\text{-}htpy}_{f\circ p} \cdot H'.$$

Of course, the type $H \cdot \mathsf{refl\text{-}htpy}_{g\circ q} \sim \mathsf{refl\text{-}htpy}_{f\circ p} \cdot H'$ is equivalent to the type $H \sim H'$, and $\sum_{(H':f\circ p\sim g\circ q)} H \sim H'$ is contractible. □

Given a cone with vertex C on a span $A \xrightarrow{f} X \xleftarrow{g} B$ and a map $h : C' \to C$, we construct a new cone with vertex C' in the following definition.

Definition 21.1.4. For any cone (p, q, H) with vertex C and any type C', we define a map

$$\mathsf{cone\text{-}map}(p, q, H) : (C' \to C) \to \mathsf{cone}(C')$$

by $h \mapsto (p \circ h, q \circ h, H \circ h)$.

Definition 21.1.5. We say that a commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$ is a **pullback square**, or that it is **cartesian**, if it satisfies the **universal property** of pullbacks, which asserts that the map

$$\mathsf{cone\text{-}map}(p, q, H) : (C' \to C) \to \mathsf{cone}(C')$$

is an equivalence for every type C'.

We often indicate the universal property with a diagram as follows:

$$\begin{tikzcd} C' \arrow[dr, dotted, "h"] \arrow[drr, bend left, "q'"] \arrow[ddr, bend right, "p'"'] & & \\ & C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ & A \arrow[r, "f"'] & X \end{tikzcd}$$

since the universal property states that for every cone (p', q', H') with vertex C', the type of pairs (h, α) consisting of $h : C' \to C$ equipped with $\alpha : \mathsf{cone\text{-}map}((p, q, H), h) = (p', q', H')$ is contractible by Theorem 8.3.6.

As a corollary we obtain the following characterization of the universal property of pullbacks.

Lemma 21.1.6. *Consider a commuting square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$ Then the following are equivalent:

(i) The square is a pullback square.

(ii) For every type C' and every cone (p', q', H') with vertex C', the type of quadruples (h, K, L, M) consisting of a map $h : C' \to C$, homotopies

$$K : p \circ h \sim p'$$
$$L : q \circ h \sim q',$$

and a homotopy $M : (H \cdot h) \cdot (g \cdot L) \sim (f \cdot K) \cdot H'$ witnessing that the square

$$\begin{tikzcd} f \circ p \circ h \arrow[r, "f \cdot K"] \arrow[d, "H \cdot h"'] & f \circ p' \arrow[d, "H'"] \\ g \circ q \circ h \arrow[r, "g \cdot L"'] & g \circ q' \end{tikzcd}$$

commutes, is contractible.

Proof. The map cone-map(p, q, H) is an equivalence if and only if its fibers are contractible. By Lemma 21.1.3 it follows that the fibers of cone-map(p, q, H) are equivalent the the described type of quadruples (h, K, L, M). □

In the following lemma we establish the uniqueness of pullbacks up to equivalence via a *3-for-2 property* for pullbacks.

Lemma 21.1.7. *Consider the squares*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd} \qquad \begin{tikzcd} C' \arrow[r, "q'"] \arrow[d, "p'"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with homotopies $H : f \circ p \sim g \circ q$ and $H' : f \circ p' \sim g \circ q'$. Furthermore, suppose we have a map $h : C' \to C$ equipped with

$$K : p \circ h \sim p'$$
$$L : q \circ h \sim q'$$
$$M : (H \cdot h) \cdot (g \cdot L) \sim (f \cdot K) \cdot H'.$$

If any two of the following three properties hold, so does the third:

(i) *C is a pullback.*

(ii) *C' is a pullback.*

(iii) *h is an equivalence.*

Proof. By the characterization of the identity type of $\mathsf{cone}(C')$ given in Lemma 21.1.3 we obtain an identification
$$\mathsf{cone\text{-}map}((p,q,H),h) = (p',q',H')$$
from the triple (K,L,M). Let D be a type, and let $k : D \to C'$ be a map. We observe that

$$\begin{aligned}
\mathsf{cone\text{-}map}((p,q,H),(h\circ k)) &\equiv (p\circ(h\circ k), q\circ(h\circ k), H\circ(h\circ k)) \\
&\equiv ((p\circ h)\circ k, (q\circ h)\circ k, (H\circ h)\circ k) \\
&\equiv \mathsf{cone\text{-}map}(\mathsf{cone\text{-}map}((p,q,H),h),k) \\
&= \mathsf{cone\text{-}map}((p',q',H'),k).
\end{aligned}$$

Thus we see that the triangle

$$\begin{tikzcd}
(D \to C') \arrow[rr, "h\circ -"] \arrow[dr, "\mathsf{cone\text{-}map}(p',q',H')"'] & & (D \to C) \arrow[dl, "\mathsf{cone\text{-}map}(p,q,H)"] \\
& \mathsf{cone}(D) &
\end{tikzcd}$$

commutes. Therefore it follows from the 3-for-2 property of equivalences established in Exercise 7.4, that if any two of the maps in this triangle is an equivalence, then so is the third. Now the claim follows from the fact that h is an equivalence if and only if $h \circ - : (D \to C') \to (D \to C)$ is an equivalence for any type D, which was established in Exercise 11.3. $\square$

Pullbacks are not only unique in the sense that any two pullbacks of the same cospan are equivalent, they are *uniquely unique* in the sense that the type of quadruples (h,K,L,M) as in Lemma 21.1.7 is contractible.

Corollary 21.1.8. *Suppose both commuting squares*

$$\begin{tikzcd}
C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
C' \arrow[r, "q'"] \arrow[d, "p'"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X
\end{tikzcd}$$

with homotopies $H : f\circ p \sim g\circ q$ and $H' : f\circ p' \sim g\circ q'$ are pullback squares. Then the type of quadruples (e,K,L,M) consisting of an equivalence $e : C' \simeq C$ equipped with

$$\begin{aligned}
K &: p\circ e \sim p' \\
L &: q\circ e \sim q' \\
M &: (H\cdot h) \cdot (g\cdot L) \sim (f\cdot K) \cdot H'.
\end{aligned}$$

is contractible.

Proof. We have seen that the type of quadruples (h, K, L, M) is equivalent to the fiber of $\mathsf{cone\text{-}map}(p, q, H)$ at (p', q', H'). By Lemma 21.1.7 it follows that h is an equivalence. Since $\mathsf{is\text{-}equiv}(h)$ is a proposition by Exercise 11.4, and hence contractible as soon as it is inhabited, it follows that the type of quadruples (e, K, L, M) is contractible. □

Corollary 21.1.9. *For any two maps $f : A \to X$ and $g : B \to X$, and any universe $\mathcal{U}$, the type*

$$\sum_{(C:\mathcal{U})}\sum_{(c:\mathsf{cone}(f,g,C))}\prod_{(C':\mathcal{U})}\mathsf{is\text{-}equiv}(\mathsf{cone\text{-}map}_{C'}(c))$$

of pullbacks in $\mathcal{U}$, is a proposition.

Proof. It is straightforward to see that the type of identifications

$$(C, (p, q, H), u) = (C', (p', q', H'), u')$$

of any two pullbacks is equivalent to the type of quadruples (e, K, L, M) as in Corollary 21.1.8. Since Corollary 21.1.8 claims that this type of quadruples is contractible, the claim follows. □

21.2 Canonical pullbacks

For every cospan we can construct a *canonical pullback*.

Definition 21.2.1. Let $f : A \to X$ and $g : B \to X$ be maps. Then we define

$$\begin{aligned}
A \times_X B &:\equiv \textstyle\sum_{(x:A)}\sum_{(y:B)} f(x) = g(y) \\
\pi_1 &:\equiv \mathsf{pr}_1 && : A \times_X B \to A \\
\pi_2 &:\equiv \mathsf{pr}_1 \circ \mathsf{pr}_2 && : A \times_X B \to B \\
\pi_3 &:\equiv \mathsf{pr}_2 \circ \mathsf{pr}_2 && : f \circ \pi_1 \sim g \circ \pi_2.
\end{aligned}$$

The type $A \times_X B$ is called the **canonical pullback** of f and g.

Note that $A \times_X B$ depends on f and g, although this dependency is not visible in the notation.

Remark 21.2.2. Given (x, y, p) and (x', y', p') in the canonical pullback $A \times_X B$, the identity type $(x, y, p) = (x', y', p')$ is equivalent to the type of triples (α, β, γ) consisting of $\alpha : x = x'$, $\beta : y = y'$, and an identification $\gamma : p \cdot \mathsf{ap}_g(\beta) = \mathsf{ap}_f(\alpha) \cdot p'$ witnessing that the square

$$\begin{tikzcd}
f(x) \arrow[r, equal, "\mathsf{ap}_f(\alpha)"] \arrow[d, equal, "p"'] & f(x') \arrow[d, equal, "p'"] \\
g(y) \arrow[r, equal, "\mathsf{ap}_g(\beta)"'] & g(y')
\end{tikzcd}$$

commutes. The proof of this fact is similar to the proof of Lemma 21.1.3.

Theorem 21.2.3. *Given maps $f : A \to X$ and $g : B \to X$, the commuting square*

$$\begin{tikzcd}
A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X,
\end{tikzcd}$$

is a pullback square.

Proof. Let C be a type. Our goal is to show that the map

$$\mathsf{cone\text{-}map}(\pi_1, \pi_2, \pi_3) : (C \to A \times_X B) \to \mathsf{cone}(C)$$

is an equivalence. Note that we have the commuting triangle

$$\begin{tikzcd}
C \to \sum_{(x:A)} \sum_{(y:B)} f(x) = g(y) \arrow[dd, "\mathsf{cone\text{-}map}"'] \arrow[dr, "\mathsf{choice}"] & \\
& \sum_{(p:C \to A)} \prod_{(z:C)} \sum_{(y:B)} f(p(z)) = g(y) \arrow[dl, "\mathsf{choice}"] \\
\sum_{(p:C \to A)} \sum_{(q:C \to B)} f \circ p \sim g \circ q. &
\end{tikzcd}$$

In this triangle the functions choice are equivalences by Theorem 11.2.1. Therefore, their composite is an equivalence. $\square$

The following corollary is now a special case of **??**, where we make sure that $f : A \to X$ and $g : B \to X$ are both maps in $\mathcal{U}$.

Corollary 21.2.4. *For any two maps $f : A \to X$ and $g : B \to X$ in $\mathcal{U}$, the type*

$$\sum_{(C:\mathcal{U})} \sum_{(c:\mathsf{cone}(f,g,C))} \prod_{(C':\mathcal{U})} \mathsf{is\text{-}equiv}(\mathsf{cone\text{-}map}_{C'}(c))$$

of pullbacks in $\mathcal{U}$, is contractible.

Definition 21.2.5. Given a commuting square

$$\begin{tikzcd}
C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X
\end{tikzcd}$$

with $H : f \circ p \sim g \circ q$, we define the **gap map**

$$\mathsf{gap}(p, q, H) : C \to A \times_X B$$

by $\lambda z.\, (p(z), q(z), H(z))$.

The following theorem provides a useful characterization of pullback squares, because in many situations it is easier to show that the gap map is an equivalence.

Theorem 21.2.6. *Consider a commuting square*

$$\begin{tikzcd}
C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X
\end{tikzcd}$$

with $H : f \circ p \sim g \circ q$. The following are equivalent:

(i) *The square is a pullback square*

(ii) *There is a term of type*

$$\mathsf{is\text{-}pullback}(p, q, H) :\equiv \mathsf{is\text{-}equiv}(\mathsf{gap}(p, q, H)).$$

Proof. Observe that we are in the situation of Lemma 21.1.7. Indeed, we have two commuting squares

$$\begin{tikzcd} A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_2"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd} \qquad \begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X, \end{tikzcd}$$

and we have the gap map $\mathsf{gap} : C \to A \times_X B$, which comes equipped with the homotopies

$$\begin{aligned} K &: \pi_1 \circ \mathsf{gap} \sim p & K &:\equiv \lambda z.\, \mathsf{refl}_{p(z)} \\ L &: \pi_2 \circ \mathsf{gap} \sim q & L &:\equiv \lambda z.\, \mathsf{refl}_{q(z)} \\ M &: (\pi_3 \cdot \mathsf{gap}) \cdot (g \cdot L) \sim (f \cdot K) \cdot H & M &:\equiv \lambda z.\, \mathsf{right\text{-}unit}(H(z)). \end{aligned}$$

Since $A \times_X B$ is shown to be a pullback in Theorem 21.2.3, it follows from Lemma 21.1.7 that C is a pullback if and only if the gap map is an equivalence. □

21.3 Cartesian products and fiberwise products as pullbacks

An important special case of pullbacks occurs when the cospan is of the form

$$A \longrightarrow \mathbf{1} \longleftarrow B.$$

In this case, the pullback is just the *cartesian product*.

Lemma 21.3.1. *Let A and B be types. Then the square*

$$\begin{tikzcd} A \times B \arrow[r, "\mathsf{pr}_2"] \arrow[d, "\mathsf{pr}_1"'] & B \arrow[d, "\mathsf{const}_\star"] \\ A \arrow[r, "\mathsf{const}_\star"'] & \mathbf{1} \end{tikzcd}$$

which commutes by the homotopy $\mathsf{const}_{\mathsf{refl}_\star}$ is a pullback square.

Proof. By Theorem 21.2.6 it suffices to show that

$$\mathsf{gap}(\mathsf{pr}_1, \mathsf{pr}_2, \lambda(a, b).\, \mathsf{refl}_\star)$$

is an equivalence. Its inverse is the map $\lambda(a, b, p).\,(a, b)$. □

The following generalization of Lemma 21.3.1 is the reason why pullbacks are sometimes called **fiber products**.

Theorem 21.3.2. *Let P and Q be families over a type X. Then the square*

$$\begin{tikzcd} \sum_{(x:X)} P(x) \times Q(x) \arrow[r, "\lambda(x,(p,q)).\,(x,q)"] \arrow[d, "\lambda(x,(p,q)).\,(x,p)"'] & \sum_{(x:X)} Q(x) \arrow[d, "\mathsf{pr}_1"] \\ \sum_{(x:X)} P(x) \arrow[r, "\mathsf{pr}_1"'] & X, \end{tikzcd}$$

which commutes by the homotopy

$$H :\equiv \lambda(x,(p,q)).\,\mathsf{refl}_x,$$

is a pullback square.

Proof. By Theorem 21.2.6 it suffices to show that the gap map is an equivalence. The gap map is homotopic to the function

$$\lambda(x,(p,q)).\,((x,p),(x,q),\mathsf{refl}_x).$$

It is easy to check that this function is an equivalence. I ts inverse is the map

$$\lambda((x,p),(y,q),\alpha).\,(y,(\mathsf{tr}_P(\alpha,p),q)).$$ □

Corollary 21.3.3. *For any $f : A \to X$ and $g : B \to X$, the square*

$$\begin{tikzcd} \sum_{(x:X)} \mathsf{fib}_f(x) \times \mathsf{fib}_g(x) \arrow[r, "\lambda(x,((a,p),(b,q))).\,b"] \arrow[d, "\lambda(x,((a,p),(b,q))).\,a"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

is a pullback square.

21.4 Fibers of maps as pullbacks

Lemma 21.4.1. *For any function $f : A \to B$, and any $b : B$, consider the square*

$$\begin{tikzcd} \mathsf{fib}_f(b) \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{pr}_1"'] & \mathbf{1} \arrow[d, "\mathsf{const}_b"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

which commutes by $\mathsf{pr}_2 : \prod_{(t:\mathsf{fib}_f(b))} f(\mathsf{pr}_1(t)) = b$. This is a pullback square.

Proof. By Theorem 21.2.6 it suffices to show that the gap map is an equivalence. The gap map is homotopic to the function

$$\mathsf{tot}(()\lambda x.\,\lambda p.\,(\star,p))$$

The map $\lambda x.\,\lambda p.\,(\star,p)$ is a family of equivalences by Exercise 8.5, so it induces an equivalence on total spaces by Theorem 9.1.3. □

Corollary 21.4.2. *For any type family B over A and any $a : A$ the square*

$$\begin{tikzcd} B(a) \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\lambda y.\,(a,y)"'] & \mathbf{1} \arrow[d, "\lambda\star.\,a"] \\ \sum_{(x:A)} B(x) \arrow[r, "\mathsf{pr}_1"'] & A \end{tikzcd}$$

is a pullback square.

Proof. To see this, note that the triangle

$$\begin{tikzcd} B(a) \arrow[rr, "\lambda b.\,((a,b),\mathsf{refl}_a)"] \arrow[dr, "\mathsf{gap}"'] & & \mathsf{fib}_{\mathsf{pr}_1}(a) \arrow[dl, "\mathsf{gap}"] \\ & \left(\sum_{(x:A)} B(x)\right) \times_A \mathbf{1}. & \end{tikzcd}$$

Since the top map is an equivalence by Exercise 8.7, and the map on the right is an equivalence by Lemma 21.4.1, it follows that the map on the left is an equivalence. The claim follows. □

21.5 Families of equivalences

Lemma 21.5.1. *Let $f : A \to B$, and let Q be a type family over B. Then the square*

$$\begin{tikzcd} \sum_{(x:A)} Q(f(x)) \arrow[r, "\lambda(x,q).\,(f(x),q)"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(y:B)} Q(b) \arrow[d, "\mathsf{pr}_1"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

commutes by $H :\equiv \lambda(x,q).\,\mathsf{refl}_{f(x)}$. This is a pullback square.

Proof. By Theorem 21.2.6 it suffices to show that the gap map is an equivalence. The gap map is homotopic to the function

$$\lambda(x,q).\,(x,(f(x),q),\mathsf{refl}_{f(x)}).$$

The inverse of this map is given by $\lambda(x,((y,q),p)).\,(x,\mathsf{tr}_Q(p^{-1},q))$, and it is straightforward to see that these maps are indeed mutual inverses. □

Theorem 21.5.2. *Let $f : A \to B$, and let $g : \prod_{(a:A)} P(a) \to Q(f(a))$ be a family of maps. The following are equivalent:*

(i) The commuting square

$$\begin{tikzcd} \sum_{(a:A)} P(a) \arrow[r, "\mathsf{tot}_f(g)"] \arrow[d, two heads] & \sum_{(b:B)} Q(b) \arrow[d, two heads] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

is a pullback square.

(ii) *g is a family of equivalences.*

Proof. The gap map is homotopic to the composite

$$\sum_{(x:A)} P(x) \xrightarrow{\mathsf{tot}(g)} \sum_{(x:A)} Q(f(x)) \xrightarrow{\mathsf{gap}'} A \times_B \Big(\sum_{(y:B)} Q(y)\Big)$$

where gap' is the gap map for the square in Lemma 21.5.1. Since gap' is an equivalence, it follows by Exercise 7.4 and Theorem 9.1.3 that the gap map is an equivalence if and only if g is a family of equivalences. □

Our goal is now to extend Theorem 21.5.2 to arbitrary pullback squares. Note that every commuting square

$$\begin{tikzcd} A \arrow[r, "h"] \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r, "i"'] & Y \end{tikzcd}$$

with $H : i \circ f \ g \circ h$ induces a map

$$\mathsf{fib\text{-}sq} : \prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(f(x))$$

on the fibers, by

$$\mathsf{fib\text{-}sq}(x, (a, p)) :\equiv (h(a), H(a)^{-1} \cdot \mathsf{ap}_i(p)).$$

Theorem 21.5.3. *Consider a commuting square*

$$\begin{tikzcd} A \arrow[r, "h"] \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r, "i"'] & Y \end{tikzcd}$$

with $H : i \circ f \ g \circ h$. The following are equivalent:

(i) *The square is a pullback square.*

(ii) *The induced map on fibers*

$$\mathsf{fib\text{-}sq} : \prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(f(x))$$

is a family of equivalences.

Proof. First we observe that the square

$$\begin{tikzcd} \sum_{(x:X)} \mathsf{fib}_f(x) \arrow[r, "\mathsf{tot}(\mathsf{fib\text{-}sq})"] \arrow[d, "\simeq"'] & \sum_{(x:X)} \mathsf{fib}_g(f(x)) \arrow[d, "\mathsf{tot}(\mathsf{tot}(\mathsf{inv}))"] \\ A \arrow[r, "\mathsf{gap}"'] & X \times_Y B \end{tikzcd}$$

commutes. To construct such a homotopy, we need to construct an identification

$$(f(a), h(a), H(a)) = (x, h(a), (H(a)^{-1} \cdot \mathsf{ap}_i(p))^{-1})$$

for every $x : X$, $a : A$, and $p : f(a) = x$. This is shown by path induction on $p : f(a) = x$. Thus, it suffices to show that

$$(f(a), h(a), H(a)) = (f(a), h(a), (H(a)^{-1} \cdot \mathsf{refl}_{i(f(a))})^{-1}),$$

which is a routine exercise.

Now we note that the left and right maps in this square are both equivalences. Therefore it follows that the top map is an equivalence if and only if the bottom map is. The claim now follows by Theorem 9.1.3. □

Corollary 21.5.4. *Consider a pullback square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X. \end{tikzcd}$$

If g is a k-truncated map, then so is p. In particular, if g is an embedding then so is p.

Proof. Since the square is assumed to be a pullback square, it follows from Theorem 21.5.3 that for each $x : A$, the fiber $\mathsf{fib}_p(x)$ is equivalent to the fiber $\mathsf{fib}_g(f(x))$, which is k-truncated. Since k-truncated types are closed under equivalences by Theorem 10.3.3, it follows that p is a k-truncated map. □

Corollary 21.5.5. *Consider a commuting square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X. \end{tikzcd}$$

and suppose that g is an equivalence. Then the following are equivalent:

(i) The square is a pullback square.

(ii) The map $p : C \to A$ is an equivalence.

Proof. If the square is a pullback square, then by Theorem 21.5.2 the fibers of p are equivalent to the fibers of g, which are contractible by Theorem 8.3.6. Thus it follows that p is a contractible map, and hence that p is an equivalence.

If p is an equivalence, then by Theorem 8.3.6 both $\mathsf{fib}_p(x)$ and $\mathsf{fib}_g(f(x))$ are contractible for any $x : X$. It follows by Exercise 8.3 that the induced map $\mathsf{fib}_p(x) \to \mathsf{fib}_g(f(x))$ is an equivalence. Thus we apply Theorem 21.5.3 to conclude that the square is a pullback. □

Theorem 21.5.6. *Consider a diagram of the form*

$$\begin{tikzcd} A \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r, "h"'] & Y. \end{tikzcd}$$

Then the type of triples (i, H, p) consisting of a map $i : A \to B$, a homotopy $H : h \circ f \sim g \circ i$, and a term p witnessing that the square

$$\begin{tikzcd} A \arrow[r, "i"] \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r, "h"'] & Y. \end{tikzcd}$$

is a pullback square, is equivalent to the type of families of equivalences

$$\prod_{(x:X)} \mathsf{fib}_f(x) \simeq \mathsf{fib}_g(h(x)).$$

Corollary 21.5.7. *Let $h : X \to Y$ be a map, and let P and Q be families over X and Y, respectively. Then the type of triples (i, H, p) consisting of a map*

$$i : \left(\sum_{(x:X)} P(x)\right) \to \left(\sum_{(y:Y)} Q(y)\right),$$

a homotopy $H : h \circ \mathsf{pr}_1 \sim \mathsf{pr}_1 \circ i$, and a term p witnessing that the square

$$\begin{tikzcd} \sum_{(x:X)} P(x) \arrow[r, "i"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(y:Y)} Q(y) \arrow[d, "\mathsf{pr}_1"] \\ X \arrow[r, "h"'] & Y. \end{tikzcd}$$

is a pullback square, is equivalent to the type of families of equivalences

$$\prod_{(x:X)} P(x) \simeq Q(h(x)).$$

One useful application of the connection between pullbacks and families of equivalences is the following theorem, which is also called the **pasting property** of pullbacks.

Theorem 21.5.8. *Consider a commuting diagram of the form*

$$\begin{tikzcd} A \arrow[r, "k"] \arrow[d, "f"'] & B \arrow[r, "l"] \arrow[d, "g"] & C \arrow[d, "h"] \\ X \arrow[r, "i"'] & Y \arrow[r, "j"'] & Z \end{tikzcd}$$

with homotopies $H : i \circ f \sim g \circ k$ and $K : j \circ g \sim h \circ l$, and the homotopy

$$(j \cdot H) \cdot (K \cdot k) : j \circ i \circ f \sim h \circ l \circ k$$

witnessing that the outer rectangle commutes. Furthermore, suppose that the square on the right is a pullback square. Then the following are equivalent:

(i) The square on the left is a pullback square.

(ii) The outer rectangle is a pullback square.

Proof. The commutativity of the two squares and the outer rectangle induces a commuting triangle

$$\begin{tikzcd}
\mathsf{fib}_f(x) \arrow[rr, "\mathsf{fib\text{-}sq}_{(f,k,H)}(x)"] \arrow[dr, "\mathsf{fib\text{-}sq}_{f,l\circ k,(j\cdot H)\cdot(K\cdot k)}(x)"'] & & \mathsf{fib}_g(i(x)) \arrow[dl, "\mathsf{fib\text{-}sq}_{(g,l,K)}(i(x))"] \\
& \mathsf{fib}_h(j(i(x))). &
\end{tikzcd}$$

A homotopy witnessing that the triangle commutes is constructed by a routine calculation.

Since the triangle commutes, and since the map $\mathsf{fib\text{-}sq}_{(g,l,K)}(i(x))$ is an equivalence for each $x : X$ by Theorem 21.5.3, it follows by the 3-for-2 property of equivalences that for each $x : X$ the top map in the triangle is an equivalence if and only if the left map is an equivalence. The claim now follows by a second application of Theorem 21.5.3. □

21.6 Descent theorems for coproducts and Σ-types

Theorem 21.6.1. *Consider maps $f : A' \to A$ and $g : B' \to B$, a map $h : X' \to X$, and commuting squares of the form*

$$\begin{tikzcd}
A' \arrow[r] \arrow[d, "f"'] & X' \arrow[d, "h"] \\
A \arrow[r] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
B' \arrow[r] \arrow[d, "g"'] & X' \arrow[d, "h"] \\
B \arrow[r] & X.
\end{tikzcd}$$

Then the following are equivalent:

(i) Both squares are pullback squares.

(ii) The commuting square

$$\begin{tikzcd}
A' + B' \arrow[r] \arrow[d, "f+g"'] & X' \arrow[d, "h"] \\
A + B \arrow[r] & X
\end{tikzcd}$$

is a pullback square.

Proof. By Theorem 21.5.3 it suffices to show that the following are equivalent:

(i) For each $x : A$ the map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_f(x) \to \mathsf{fib}_h(\alpha_A(x))$$

is an equivalence, and for each $y : B$ the map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_g(y) \to \mathsf{fib}_h(\alpha_B(y))$$

is an equivalence.

(ii) For each $t : A + B$ the map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_{f+g}(t) \to \mathsf{fib}_h(\alpha(t))$$

is an equivalence.

By the dependent universal property of coproducts, the second claim is equivalent to the claim that both for each $x : A$ the map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_{f+g}(\mathsf{inl}(x)) \to \mathsf{fib}_h(\alpha_A(x))$$

is an equivalence, and for each $y : B$, the map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_{f+g}(\mathsf{inr}(y)) \to \mathsf{fib}_h(\alpha_B(y))$$

is an equivalence.

We claim that there is a commuting triangle

$$\begin{tikzcd} \mathsf{fib}_f(x) \arrow[rr] \arrow[dr] & & \mathsf{fib}_{f+g}(\mathsf{inl}(x)) \arrow[dl] \\ & \mathsf{fib}_h(\alpha_A(x)) & \end{tikzcd}$$

for every $x : A$. To see that the triangle commutes, we need to construct an identification

The top map is given by

$$(a', p) \mapsto (\mathsf{inl}(a'), \mathsf{ap}_{\mathsf{inl}}(p)).$$

The triangle then commutes by the homotopy

$$(a', p) \mapsto \mathsf{eq\text{-}pair}(\mathsf{refl}, \mathsf{ap}_{\mathsf{concat}(H(a')^{-1})}(\mathsf{ap\text{-}comp}_{[\alpha_A,\alpha_B],inl}))$$

We note that the top map is an equivalence, so it follows by the 3-for-2 property of equivalences that the left map is an equivalence if and only if the right map is an equivalence.

Similarly, there is a commuting triangle

$$\begin{tikzcd} \mathsf{fib}_g(y) \arrow[rr] \arrow[dr] & & \mathsf{fib}_{f+g}(\mathsf{inr}(y)) \arrow[dl] \\ & \mathsf{fib}_h(\alpha_B(y)) & \end{tikzcd}$$

in which the top map is an equivalence, completing the proof. □

In the following corollary we conclude that coproducts distribute over pullbacks.

Corollary 21.6.2. *Consider a cospan of the form*

$$\begin{tikzcd} & Y \arrow[d] \\ A + B \arrow[r] & X. \end{tikzcd}$$

Then there is an equivalence

$$(A + B) \times_X Y \simeq (A \times_X Y) + (B \times_X Y).$$

Theorem 21.6.3. *Consider a family of maps $f_i : A'_i \to A_i$ indexed by a type I, a map $h : X' \to X$, and a commuting square*

$$\begin{tikzcd} A'_i \arrow[r] \arrow[d, "f_i"'] & X' \arrow[d, "h"] \\ A_i \arrow[r, "\alpha_i"'] & X \end{tikzcd}$$

for each $i : I$. Then the following are equivalent:

(i) For each $i : I$ the square is a pullback square.

(ii) The commuting square

$$\begin{tikzcd} \sum_{(i:I)} A'_i \arrow[r] \arrow[d, "\mathsf{tot}(f)"'] & X' \arrow[d, "h"] \\ \sum_{(i:I)} A_i \arrow[r, "\alpha"] & X \end{tikzcd}$$

is a pullback square.

Proof. By Theorem 21.5.3 it suffices to show that the following are equivalent for each $i : I$ and $a : A_i$:

(i) The map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_{f_i}(a) \to \mathsf{fib}_g(\alpha_i(a))$$

is an equivalence.

(ii) The map

$$\mathsf{fib\text{-}sq} : \mathsf{fib}_{\mathsf{tot}(f)}(i, a) \to \mathsf{fib}_g(\alpha_i(a))$$

is an equivalence.

To see this, note that we have a commuting triangle

$$\begin{tikzcd} \mathsf{fib}_{f_i}(a) \arrow[rr] \arrow[dr] & & \mathsf{fib}_{\mathsf{tot}(f)}(i, a) \arrow[dl] \\ & \mathsf{fib}_g(\alpha_i(a)), & \end{tikzcd}$$

where the top map is an equivalence by Lemma 9.1.2. Therefore the claim follows by the 3-for-2 property of equivalences. □

In the following corollary we conclude that Σ distributes over coproducts.

Corollary 21.6.4. *Consider a cospan of the form*

$$\begin{tikzcd} & Y \arrow[d] \\ \sum_{(i:I)} A_i \arrow[r] & X. \end{tikzcd}$$

Then there is an equivalence

$$\left(\sum_{(i:I)} A_i\right) \times_X Y \simeq \sum_{(i:I)} (A_i \times_X Y).$$

Exercises

21.1 (a) Show that the square

$$\begin{tikzcd} (x=y) \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{const}_y"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_x"'] & A \end{tikzcd}$$

is a pullback square.

(b) Show that the square

$$\begin{tikzcd} (x=y) \arrow[r, "\mathsf{const}_x"] \arrow[d, "\mathsf{const}_\star"'] & A \arrow[d, "\delta_A"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_{(x,y)}"'] & A \times A \end{tikzcd}$$

is a pullback square, where $\delta_A : A \to A \times A$ is the diagonal of A, defined in Exercise 10.2.

21.2 In this exercise we give an alternative characterization of the notion of k-truncated map, compared to Theorem 10.3.6. Given a map $f : A \to X$ define the **diagonal** of f to be the map $\delta_f : A \to A \times_X A$ given by $x \mapsto (x, x, \mathsf{refl}_{f(x)})$.

(a) Construct an equivalence

$$\mathsf{fib}_{\delta_f}((x,y,p)) \simeq \mathsf{fib}_{\mathsf{ap}_f}(p)$$

to show that the square

$$\begin{tikzcd} \mathsf{fib}_{\mathsf{ap}_f}(p) \arrow[r, "\mathsf{const}_x"] \arrow[d, "\mathsf{const}_\star"'] & A \arrow[d, "\delta_f"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_{(x,y,p)}"'] & A \times_X A \end{tikzcd}$$

is a pullback square, for every $x, y : A$ and $p : f(x) = f(y)$.

(b) Show that a map $f : A \to X$ is $(k+1)$-truncated if and only if δ_f is k-truncated.

Conclude that f is an embedding if and only if δ_f is an equivalence.

21.3 Consider a commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$. Show that this square is a pullback square if and only if the square

$$\begin{tikzcd} C \arrow[r, "p"] \arrow[d, "q"'] & A \arrow[d, "f"] \\ B \arrow[r, "g"'] & X \end{tikzcd}$$

with $H^{-1} : g \circ q \sim f \circ p$ is a pullback square.

21.4 Show that any square of the form

$$\begin{tikzcd} C \arrow[r] \arrow[d] & B \arrow[d] \\ \emptyset \arrow[r] & X \end{tikzcd}$$

commutes and is a pullback square. This is the *descent property* of the empty type.

21.5 Consider a commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$. Show that the following are equivalent:

(i) The square is a pullback square.

(ii) For every type T, the commuting square

$$\begin{tikzcd} C^T \arrow[r, "q\circ -"] \arrow[d, "p\circ -"'] & B^T \arrow[d, "g\circ -"] \\ A^T \arrow[r, "f\circ -"'] & X^T \end{tikzcd}$$

is a pullback square.

Note: property (ii) is really just a rephrasing of the universal property of pullbacks.

21.6 Consider a commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$. Show that the following are equivalent:

(i) The square is a pullback square.

(ii) The square

$$\begin{tikzcd} C \arrow[r, "g\circ q"] \arrow[d, "\lambda x.\,(p(x),q(x))"'] & X \arrow[d, "\delta_X"] \\ A \times B \arrow[r, "f\times g"'] & X \times X \end{tikzcd}$$

which commutes by $\lambda z.\,\mathsf{eq\text{-}pair}(H(z), \mathsf{refl}_{g(q(z))})$ is a pullback square.

21.7 Consider two commuting squares

$$\begin{tikzcd} C_1 \arrow[r] \arrow[d] & B_1 \arrow[d] \\ A_1 \arrow[r] & X_1 \end{tikzcd} \qquad \begin{tikzcd} C_2 \arrow[r] \arrow[d] & B_2 \arrow[d] \\ A_2 \arrow[r] & X_2. \end{tikzcd}$$

(a) Show that if both squares are pullback squares, then the square

$$\begin{tikzcd} C_1 \times C_2 \arrow[r] \arrow[d] & B_1 \times B_2 \arrow[d] \\ A_1 \times A_2 \arrow[r] & X_1 \times X_2. \end{tikzcd}$$

is also a pullback square.

(b) Show that if there are terms $t_1 : A_1 \times_{X_1} B_1$ and $t_2 : A_2 \times_{X_2} B_2$, then the converse of (a) also holds.

21.8 Consider for each $i : I$ a pullback square

$$\begin{tikzcd} C_i \arrow[r, "q_i"] \arrow[d, "p_i"'] & B_i \arrow[d, "g_i"] \\ A_i \arrow[r, "f_i"'] & X_i \end{tikzcd}$$

with $H_i : f_i \circ p_i \sim g_i \circ q_i$. Show that the commuting square

$$\begin{tikzcd} \prod_{(i:I)} C_i \arrow[r] \arrow[d] & \prod_{(i:I)} B_i \arrow[d] \\ \prod_{(i:I)} A_i \arrow[r] & \prod_{(i:I)} X_i \end{tikzcd}$$

is a pullback square.

21.9 Let $f : A \to B$ be a map. Show that the following are equivalent:

(i) The commuting square

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & \|A\| \arrow[d, "\|f\|"] \\ B \arrow[r] & \|B\|. \end{tikzcd}$$

is a pullback square.

(ii) There is a term of type $A \to \mathsf{is\text{-}equiv}(f)$.

(iii) The commuting square

$$\begin{tikzcd} A \times A \arrow[r, "f \times f"] \arrow[d, "\mathsf{pr}_1"'] & B \times B \arrow[d, "\mathsf{pr}_1"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

is a pullback square.

21.10 Consider a pullback square

$$\begin{tikzcd} A' \arrow[r, "p"] \arrow[d, "f'"'] & A \arrow[d, "f"] \\ B' \arrow[r, "q"'] & B, \end{tikzcd}$$

in which $q : B' \to B$ is surjective. Show that if $f' : A' \to B'$ is an embedding, then so is $f : A \to B$.

21.11 Consider a family of diagrams of the form

$$\begin{tikzcd} A_i \arrow[r] \arrow[d, "f_i"'] & C \arrow[r] \arrow[d, "g"] & X \arrow[d, "h"] \\ B_i \arrow[r] & D \arrow[r] & Y \end{tikzcd}$$

indexed by $i : I$, in which the left squares are pullback squares, and assume that the induced map

$$\left(\sum_{(i:I)} B_i\right) \to D$$

is surjective. Show that the following are equivalent:

(i) For each $i : I$ the outer rectangle is a pullback square.

(ii) The right square is a pullback square.

Hint: By Theorem 21.6.3 it suffices to prove this equivalence for a single diagram of the form

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & C \arrow[r] \arrow[d, "g"'] & X \arrow[d, "h"] \\ B \arrow[r] & D \arrow[r] & Y \end{tikzcd}$$

where the map $B \to D$ is assumed to be surjective.

21.12 Consider a pullback square

$$\begin{tikzcd} E' \arrow[r, "g"] \arrow[d, "p'"'] & E \arrow[d, "p"] \\ B' \arrow[r, "f"'] & B \end{tikzcd}$$

in which p is assumed to be surjective. Show that p' is also surjective, and show that the following are equivalent:

(i) The map f is an equivalence.

(ii) The map g is an equivalence.

21.13 Show that a map $f : A \to B$ is an equivalence if and only if the square

$$\begin{tikzcd} A^B \arrow[r, "f\circ -"] \arrow[d, "-\circ f"'] & B^B \arrow[d, "-\circ f"] \\ A^A \arrow[r, "f\circ -"'] & B^A \end{tikzcd}$$

is a pullback square.

22 Homotopy pushouts

A common way in topology to construct new spaces is by attaching cells to a given space. A 0-cell is just a point, an 1-cell is an interval, a 2-cell is a disc, a 3-cell is the solid ball, and so forth. Many spaces can be obtained by attaching cells. For example, the circle is obtained by attaching a 1-cell to a 0-cell, so that both end-points of the interval are mapped to the point. More generally, an n-sphere is obtained by attaching an n-disc to the point, so that its entire boundary gets mapped to the point.

In type theory we can also consider a notion of n-cells. Just as in topology, a 0-cell is just a point (i.e., a term). A 1-cell, however, is in type theory an identification, i.e., a term of the identity type. A 1-cell is then an identification of identifications, and so forth. Then we can attach cells to a type by taking a pushout, which is a process dual to taking a pullback.

The idea of pushouts is to glue two types A and B together using a mediating type S and maps $f : S \to A$ and $g : S \to B$. In other words, we start with a diagram of the form

$$A \xleftarrow{f} S \xrightarrow{g} B.$$

We call such a triple $\mathcal{S} \equiv (S, f, g)$ a **span** from A to B. A span from A to B can be thought of as a relation from A to B, relating $f(s)$ to $g(s)$ for any $s : S$. The pushout of the span $\mathcal{S}$ is then a type X that comes equipped with inclusion maps $i : A \to X$ and $j : B \to X$ and a homotopy H witnessing that the square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

Note that this homotopy makes sure that there is a path $H(s) : i(f(s)) = j(g(s))$ for every $s : S$. In other words, any $x : A$ and $y : B$ that are related by in $\mathcal{S}$ become identified in the pushout. The last requirement of the pushout is that it satisfies a universal property that is dual to the universal property of pullbacks.

There are several equivalent characterizations of pushouts. Two such characterizations are studied in this section, establishing the duality between pullbacks and pushouts. Other characterizations, including the induction principle of pushouts, and the *dependent universal property* of pushouts, are studied in §24.

Unlike pullbacks, however, it is not automatically the case that pushouts always exist. We will therefore postulate as an axiom that pushouts always exist. Moreover, we will assume that universes are closed under pushouts.

22.1 The universal property of pushouts

Definition 22.1.1. Consider a span $\mathcal{S} \equiv (S, f, g)$ from A to B, and let X be a type. A **cocone** with vertex X on $\mathcal{S}$ is a triple (i, j, H) consisting of maps $i : A \to X$ and $j : B \to X$, and a homotopy $H : i \circ f \sim j \circ g$ witnessing that the square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

commutes. We write $\mathsf{cocone}_{\mathcal{S}}(X)$ for the type of cocones on $\mathcal{S}$ with vertex X.

Remark 22.1.2. Given two cocones (i, j, H) and (i', j', H') with vertex X, the type of identifications $(i, j, H) = (i', j', H')$ in $\mathsf{cocone}_{\mathcal{S}}(X)$ is equivalent to the type of triples (K, L, M) consisting of

$$K : i \sim i'$$

$$L : j \sim j',$$

and a homotopy M witnessing that the square

$$\begin{tikzcd} i \circ f \arrow[r, "K \cdot f"] \arrow[d, "H"'] & i' \circ f \arrow[d, "H'"] \\ j \circ g \arrow[r, "L \cdot g"'] & j' \circ g \end{tikzcd}$$

of homotopies commutes.

Definition 22.1.3. Consider a cocone (i, j, H) with vertex X on the span $\mathcal{S} \equiv (S, f, g)$, as indicated in the following commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

For every type Y, we define the map

$$\mathsf{cocone\text{-}map}(i, j, H) : (X \to Y) \to \mathsf{cocone}(Y)$$

by $h \mapsto (h \circ i, h \circ j, h \cdot H)$.

Definition 22.1.4. A commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$ is said to be a **(homotopy) pushout square** if the cocone (i, j, H) with vertex X on the span $\mathcal{S} \equiv (S, f, g)$ satisfies the **universal property of pushouts**, which asserts that the map

$$\mathsf{cocone\text{-}map}(i, j, H) : (X \to Y) \to \mathsf{cocone}(Y)$$

is an equivalence for any type Y. Sometimes pushout squares are also called **cocartesian squares**.

Lemma 22.1.5. *Consider a pushout square*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, and consider a commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j'"] \\ A \arrow[r, "i'"'] & X'. \end{tikzcd}$$

with $H' : i' \circ f \sim j' \circ g$. Then the type of maps $h : X \to X'$ equipped with homotopies

$$K : h \circ i \sim i'$$
$$L : h \circ j \sim j'$$

and a homotopy M witnessing that the square

$$\begin{tikzcd} h \circ i \circ f \arrow[r, "K \cdot f"] \arrow[d, "h \cdot H"'] & i' \circ f \arrow[d, "H'"] \\ h \circ j \circ g \arrow[r, "L \cdot g"'] & j' \circ g \end{tikzcd}$$

commutes, is contractible.

Proof. For any map $h : X \to X'$, the type of triples (K, L, M) as in the statement of the lemma is equivalent to the type of identifications

$$\text{cocone-map}((i, j, H), h) = (i', j', H'),$$

by Remark 22.1.2. Therefore it follows that the type of quadruples (h, K, L, M) is equivalent to the fiber of $\text{cocone-map}(i, j, H)$ at (i', j', H'). Since we have assumed that the cocone (i, j, H) satisfies the universal property of the pushout of $\mathcal{S}$, the map $\text{cocone-map}(i, j, H)$ is an equivalence, and therefore it has contractible fibers by Theorem 8.3.6. □

Theorem 22.1.6. *Consider two cocones*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd} \qquad \begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j'"] \\ A \arrow[r, "i'"'] & X' \end{tikzcd}$$

on a span $\mathcal{S} \equiv (S, f, g)$, and let $h : X \to X'$ be a map equipped with homotopies

$$K : h \circ i \sim i'$$
$$L : h \circ j \sim j'$$

and a homotopy M witnessing that the square

$$\begin{tikzcd} h \circ i \circ f \arrow[r, "K \cdot f"] \arrow[d, "h \cdot H"'] & i' \circ f \arrow[d, "H'"] \\ h \circ j \circ g \arrow[r, "L \cdot g"'] & j' \circ g \end{tikzcd}$$

commutes. Then if any two of the following three statements hold, so does the third:

(i) The cocone (i, j, H) satisfies the universal property of the pushout of $\mathcal{S}$.

(ii) The cocone (i', j', H') satisfies the universal property of the pushotu of $\mathcal{S}$.

(iii) The map h is an equivalence.

Proof. First we observe that we have a commuting triangle

$$\begin{tikzcd} (X' \to Y) \arrow[rr, "- \circ h"] \arrow[dr, "\mathsf{cocone\text{-}map}(i',j',H')"'] & & (X \to Y) \arrow[dl, "\mathsf{cocone\text{-}map}(i,j,H)"] \\ & \mathsf{cocone}_{\mathcal{S}}(Y) & \end{tikzcd}$$

for any type Y. Therefore it follows from the 3-for-2 property of equivalences that if any two of the maps in this triangle is an equivalence, so is the third. Now the claim follows from the observation in Theorem 11.4.1 that h is an equivalence if and only if the map $- \circ h : (X' \to Y) \to (X \to Y)$ is an equivalence for any type Y. $\square$

In the following corollary we establish the fact that pushouts are *uniquely unique*.

Corollary 22.1.7. *Consider two pushouts*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd} \qquad \begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j'"] \\ A \arrow[r, "i'"'] & X' \end{tikzcd}$$

of a given span $\mathcal{S} \equiv (S, f, g)$. Then the type of equivalences $e : X \simeq X'$ equipped with homotopies

$$K : h \circ i \sim i'$$
$$L : h \circ j \sim j'$$

and a homotopy M witnessing that the square

$$\begin{tikzcd} h \circ i \circ f \arrow[r, "K \cdot f"] \arrow[d, "h \cdot H"'] & i' \circ f \arrow[d, "H'"] \\ h \circ j \circ g \arrow[r, "L \cdot g"'] & j' \circ g \end{tikzcd}$$

commutes, is contractible.

Proof. This follows from combining Lemma 22.1.5 and Theorem 22.1.6. $\square$

Corollary 22.1.8. *Consider a span*

$$A \xleftarrow{f} S \xrightarrow{g} B$$

in a universe $\mathcal{U}$. Then the type

$$\sum_{(X:\mathcal{U})} \sum_{(c:\mathsf{cocone}(X))} \prod_{(Y:\mathcal{U})} \mathsf{is\text{-}equiv}(\mathsf{cocone\text{-}map}_Y(c))$$

of is a proposition.

Proof. It is routine to verify that the type of quadruples (e, K, L, M) as in Corollary 22.1.8 is equivalent to the identity type of the type of pushouts of the span $\mathcal{S} \equiv (S, f, g)$. The claim then follows, since Corollary 22.1.8 asserts that this type of quadruples is contractible. $\square$

22.2 Suspensions

A particularly important class of examples of pushouts are suspensions.

Definition 22.2.1. Let X be a type. A **suspension** of X is a type ΣX equipped with a **north pole** $\mathsf{N} : \Sigma X$, a **south pole** $\mathsf{S} : \Sigma X$, and a **meridian**

$$\mathsf{merid} : X \to (\mathsf{N} = \mathsf{S}),$$

such that the commuting square

$$\begin{tikzcd} X \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{const}_\star"'] & \mathbf{1} \arrow[d, "\mathsf{const}_\mathsf{S}"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_\mathsf{N}"'] & \Sigma X \end{tikzcd}$$

is a pushout square.

We can use suspensions to present the spheres in type theory. The 2-sphere is a space which, like the surface of the earth, has a north pole and a south pole. Moreover, for each point of the equator there is a meridian that connects the north pole to the south pool. Of course, the equator is a circle, so we see that the 2-sphere is just the suspension of the circle.

Similarly we can see that the $(n+1)$-sphere must be the suspension of the n-sphere. The $(n+1)$-sphere is the unit sphere in the vector space $\mathbb{R}^{n+2}$. This vector space has an orthogonal basis $e_1, \ldots, e_{n+2}$. Then the north and the south pole are given by e_{n+2} and $-e_{n+2}$, respectively, and for each unit vector in $\mathbb{R}^{n+1} \subseteq \mathbb{R}^{n+2}$ we have a meridian connecting the north pole with the south pole. The unit sphere in $\mathbb{R}^{n+1}$ is of course the n-sphere, so we see that the $(n+1)$-sphere must be a suspension of the n-sphere.

These observations suggest that we can define the spheres by recursion on n. Note that the spheres in type theory are defined entirely synthetically, i.e., without reference to the ambient topological space $\mathbb{R}^{n+1}$. Indeed, from a homotopical point of view each space $\mathbb{R}^n$ is contractible, so in type theory it is just presented as the unit type[1].

Definition 22.2.2. We define the n**-sphere** $\mathbf{S}^n$ for any $n : \mathbb{N}$ by induction on n, by taking

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

Remark 22.2.3. Note that this recursive definition of the spheres only goes through in type theory if we have (or assume) a universe that is closed under suspensions.

In the following lemma we give a slight simplification of the universal property of suspensions, making it just a little easier to work with them.

Lemma 22.2.4. *Let X and Y be types, and let ΣX be a suspension of X. Then the map*

$$(\Sigma X \to Y) \to \textstyle\sum_{(y,y':Y)} X \to (y = y')$$

given by $f \mapsto (f(\mathsf{N}), f(\mathsf{S}), f \cdot \mathsf{merid})$ is an equivalence.

[1]It is an entirely different matter to define the *set* $\mathbb{R}$ rather than the homotopy type of $\mathbb{R}$. See Chapter 11 of [3] for definitions of the Dedekind reals and the Cauchy reals.

Proof. Note that we have a commuting triangle

$$\begin{tikzcd} & (\Sigma X \to Y) \arrow[dl, "\mathsf{cocone\text{-}map}"'] \arrow[dr, "f \mapsto (f(\mathsf{N}), f(\mathsf{S}), f \cdot \mathsf{merid})"] & \\ \mathsf{cocone}_{\mathcal{S}}(Y) \arrow[rr] & & \sum_{(y,y':Y)} X \to (y = y') \end{tikzcd}$$

where $\mathcal{S}$ is the span $\mathbf{1} \leftarrow X \rightarrow \mathbf{1}$. The bottom map is given by $(i,j,H) \mapsto (i(\star), j(\star), H)$. This map is an equivalence, and the map on the left is an equivalence by the assumption that ΣX is a suspension of X. Therefore the claim follows by the 3-for-2 property of equivalences. ☐

22.3 The duality of pullbacks and pushouts

Lemma 22.3.1. *For any span $\mathcal{S} \equiv (S, f, g)$ from A to B, and any type X the square*

$$\begin{tikzcd} \mathsf{cocone}_{\mathcal{S}}(X) \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & X^B \arrow[d, "-\circ g"] \\ X^A \arrow[r, "-\circ f"'] & X^S, \end{tikzcd}$$

which commutes by the homotopy $\pi_3' :\equiv \lambda(i,j,H).\, \mathsf{eq\text{-}htpy}(H)$, is a pullback square.

Proof. The gap map $\mathsf{cocone}_{\mathcal{S}}(X) \to X^A \times_{X^S} X^B$ is the function

$$\lambda(i,j,H).\,(i,j,\mathsf{eq\text{-}htpy}(H)).$$

This is an equivalence by Theorem 9.1.3, since it is the induced map on total spaces of the family of equivalences $\mathsf{eq\text{-}htpy}$. Therefore, the square is a pullback square by Theorem 21.2.6. ☐

In the following theorem we establish the duality between pullbacks and pushouts.

Theorem 22.3.2. *Consider a commuting square*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X, \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$. The following are equivalent:

(i) The square is a pushout square.

(ii) The square

$$\begin{tikzcd} T^X \arrow[r, "-\circ j"] \arrow[d, "-\circ i"'] & T^B \arrow[d, "-\circ g"] \\ T^A \arrow[r, "-\circ f"'] & T^S \end{tikzcd}$$

which commutes by the homotopy

$$\lambda h.\, \mathsf{eq\text{-}htpy}(h \cdot H)$$

is a pullback square, for every type T.

Proof. It is straightforward to verify that the triangle

$$\begin{tikzcd} & T^X \arrow[dl, "\mathsf{cocone\text{-}map}(i,j,H)"'] \arrow[dr, "\mathsf{gap}(-\circ i,-\circ j,\mathsf{eq\text{-}htpy}(-\cdot H))"] & \\ \mathsf{cocone}(T) \arrow[rr, "\mathsf{gap}(i,j,\mathsf{eq\text{-}htpy}(H))"'] & & T^A \times_{T^S} T^B \end{tikzcd}$$

commutes. Since the bottom map is an equivalence by Lemma 22.3.1, it follows that if either one of the remaining maps is an equivalence, so is the other. The claim now follows by Theorem 21.2.6. □

Example 22.3.3. The square

$$\begin{tikzcd} X^{\mathbf{S}^1} \arrow[r, "-\circ\mathsf{const}_{\mathsf{base}}"] \arrow[d, "-\circ\mathsf{const}_{\mathsf{base}}"'] & X^{\mathbf{1}} \arrow[d, "-\circ\mathsf{const}_{\star}"] \\ X^{\mathbf{1}} \arrow[r, "-\circ\mathsf{const}_{\star}"'] & X^{\mathbf{2}} \end{tikzcd}$$

is a pullback square for each type X. Therefore it follows by the second characterization of pushouts in Theorem 22.3.2 that the circle is a pushout

$$\begin{tikzcd} \mathbf{2} \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \mathbf{1} \arrow[r] & \mathbf{S}^1. \end{tikzcd}$$

In other words, $\mathbf{S}^1 \simeq \Sigma\mathbf{2}$.

Theorem 22.3.4. *Consider the following configuration of commuting squares:*

$$\begin{tikzcd} A \arrow[r, "i"] \arrow[d, "f"'] & B \arrow[r, "k"] \arrow[d, "g"'] & C \arrow[d, "h"] \\ X \arrow[r, "j"'] & Y \arrow[r, "l"'] & Z \end{tikzcd}$$

with homotopies $H : j \circ f \sim g \circ i$ and $K : l \circ g \sim h \circ k$, and suppose that the square on the left is a pushout square. Then the square on the right is a pushout square if and only if the outer rectangle is a pushout square.

Proof. Let T be a type. Taking the exponent $T^{(-)}$ of the entire diagram of the statement of the theorem, we obtain the following commuting diagram

$$\begin{tikzcd} T^Z \arrow[r, "-\circ l"] \arrow[d, "-\circ h"'] & T^Y \arrow[r, "-\circ j"] \arrow[d, "-\circ g"'] & T^X \arrow[d, "-\circ f"] \\ T^C \arrow[r, "-\circ k"'] & T^B \arrow[r, "-\circ i"'] & T^A. \end{tikzcd}$$

By the assumption that Y is the pushout of $B \leftarrow A \rightarrow X$, it follows that the square on the right is a pullback square. It follows by Theorem 21.5.8 that the rectangle on the left is a pullback if and only if the outer rectangle is a pullback. Thus the statement follows by the second characterization in Theorem 22.3.2. □

Lemma 22.3.5. *Consider a map $f : A \to B$. Then the cofiber of the map* $\mathsf{inr} : B \to \mathsf{cofib}_f$ *is equivalent to the suspension ΣA of A.*

22.4 Fiber sequences and cofiber sequences

Definition 22.4.1. Given a map $f : A \to B$, we define the **cofiber** cofib_f of f as the pushout

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d] & B \arrow[d, "\mathsf{inr}"] \\ \mathbf{1} \arrow[r, "\mathsf{inl}"'] & \mathsf{cofib}_f. \end{tikzcd}$$

The cofiber of a map is sometimes also called the **mapping cone**.

Example 22.4.2. The suspension ΣX of X is the cofiber of the map $X \to \mathbf{1}$.

22.5 Further examples of pushouts

Definition 22.5.1. We define the **join** $X * Y$ of X and Y to be the pushout

$$\begin{tikzcd} X \times Y \arrow[r, "\mathsf{pr}_2"] \arrow[d, "\mathsf{pr}_1"'] & Y \arrow[d, "\mathsf{inr}"] \\ X \arrow[r, "\mathsf{inl}"'] & X * Y. \end{tikzcd}$$

Definition 22.5.2. Suppose A and B are pointed types, with base points a_0 and b_0, respectively. The **(binary) wedge** $A \vee B$ of A and B is defined as the pushout

$$\begin{tikzcd} \mathbf{2} \arrow[r] \arrow[d] & A + B \arrow[d] \\ \mathbf{1} \arrow[r] & A \vee B. \end{tikzcd}$$

Definition 22.5.3. Given a type I, and a family of pointed types A over i, with base points $a_0(i)$. We define the **(indexed) wedge** $\bigvee_{(i:I)} A_i$ as the pushout

$$\begin{tikzcd} I \arrow[r, "\lambda i.\,(i, a_0(i))"] \arrow[d] & \sum_{(i:I)} A_i \arrow[d] \\ \mathbf{1} \arrow[r] & \bigvee_{(i:I)} A_i. \end{tikzcd}$$

Definition 22.5.4. Let X and Y be types with base points x_0 and y_0, respectively. We define the **wedge** $X \vee Y$ of X and Y to be the pushout

$$\begin{tikzcd} \mathbf{2} \arrow[r, "\mathsf{ind}_{\mathbf{2}}(\mathsf{inl}(x_0), \mathsf{inr}(y_0))"] \arrow[d, "\mathsf{const}_\star"'] & X + Y \arrow[d, "\mathsf{inr}"] \\ \mathbf{1} \arrow[r, "\mathsf{inl}"'] & X \vee Y \end{tikzcd}$$

Definition 22.5.5. Let X and Y be types with base points x_0 and y_0, respectively. We define a map

$$\mathsf{wedge\text{-}incl} : X \vee Y \to X \times Y.$$

as the unique map obtained from the commutative square

$$\begin{tikzcd} \mathbf{2} \arrow[r, "\mathsf{ind}_{\mathbf{2}}(\mathsf{inl}(x_0),\mathsf{inr}(y_0))"] \arrow[d, "\mathsf{const}_\star"'] & X+Y \arrow[d, "\mathsf{ind}_{X+Y}(\lambda x.\,(x,y_0),\lambda y.\,(x_0,y))"] \\ \mathbf{1} \arrow[r, "\lambda t.\,(x_0,y_0)"'] & X \times Y. \end{tikzcd}$$

Definition 22.5.6. We define the **smash product** $X \wedge Y$ of X and Y to be the pushout

$$\begin{tikzcd} X \vee Y \arrow[r, "\mathsf{wedge\text{-}incl}"] \arrow[d, "\mathsf{const}_\star"'] & X \times Y \arrow[d, "\mathsf{inr}"] \\ \mathbf{1} \arrow[r, "\mathsf{inl}"'] & X \wedge Y. \end{tikzcd}$$

Exercises

22.1 Use Theorems 11.4.1 and 22.3.2 and Corollary 21.5.5 to show that for any commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"', "\simeq"] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & C \end{tikzcd}$$

where f is an equivalence, the square is a pushout square if and only if $j : B \to C$ is an equivalence. Use this observation to conclude the following:

(i) If X is contractible, then ΣX is contractible.

(ii) The cofiber of any equivalence is contractible.

(iii) The cofiber of a point in B (i.e., of a map of the type $\mathbf{1} \to B$) is equivalent to B.

(iv) There is an equivalence $X \simeq \emptyset * X$.

(v) If X is contractible, then $X * Y$ is contractible.

(vi) If A is contractible, then there is an equivalence $A \vee B \simeq B$ for any pointed type B.

22.2 Let P and Q be propositions.

(a) Show that $P * Q$ satisfies the *universal property of disjunction*, i.e., that for any proposition R, the map

$$(P * Q \to R) \to (P \to R) \times (Q \to R)$$

given by $f \mapsto (f \circ \mathsf{inl}, f \circ \mathsf{inr})$, is an equivalence.

(b) Use the proposition $R :\equiv \mathsf{is\text{-}contr}(P * Q)$ to show that $P * Q$ is again a proposition.

22.3 Let Q be a proposition, and let A be a type. Show that the following are equivalent:

(i) The map $(Q \to A) \to (\emptyset \to A)$ is an equivalence.

(ii) The type A^Q is contractible.

(iii) There is a term of type $Q \to \mathsf{is\text{-}contr}(A)$.

(iv) The map $\mathsf{inr} : A \to Q * A$ is an equivalence.

22.4 Let P be a proposition. Show that ΣP is a set, with an equivalence

$$\big(\mathsf{inl}(\star) = \mathsf{inr}(\star)\big) \simeq P.$$

22.5 Show that $A \sqcup^{\mathcal{S}} B \simeq B \sqcup^{\mathcal{S}^{\mathsf{op}}} A$, where $\mathcal{S}^{\mathsf{op}} :\equiv (S, g, f)$ is the **opposite span** of $\mathcal{S}$.

22.6 Use Exercise 21.8 to show that if

$$\begin{tikzcd} S \arrow[r] \arrow[d] & Y \arrow[d] \\ X \arrow[r] & Z \end{tikzcd}$$

is a pushout square, then so is

$$\begin{tikzcd} A \times S \arrow[r] \arrow[d] & A \times Y \arrow[d] \\ A \times X \arrow[r] & A \times Z \end{tikzcd}$$

for any type A.

22.7 Use Exercise 21.7 to show that if

$$\begin{tikzcd} S_1 \arrow[r] \arrow[d] & Y_1 \arrow[d] \\ X_1 \arrow[r] & Z_1 \end{tikzcd} \qquad \begin{tikzcd} S_2 \arrow[r] \arrow[d] & Y_2 \arrow[d] \\ X_2 \arrow[r] & Z_2 \end{tikzcd}$$

are pushout squares, then so is

$$\begin{tikzcd} S_1 + S_2 \arrow[r] \arrow[d] & Y_1 + Y_2 \arrow[d] \\ X_1 + X_2 \arrow[r] & Z_1 + Z_2. \end{tikzcd}$$

22.8 (a) Consider a span (S, f, g) from A to B. Use Exercise 21.6 to show that the square

$$\begin{tikzcd} S + S \arrow[r, "{[\mathsf{id},\mathsf{id}]}"] \arrow[d, "f+g"'] & S \arrow[d, "\mathsf{inr} \circ g"] \\ A + B \arrow[r, "{[\mathsf{inl},\mathsf{inr}]}"'] & A \sqcup^{\mathcal{S}} B \end{tikzcd}$$

is again a pushout square.

(b) Show that $\Sigma X \simeq \mathbf{2} * X$.

22.9 Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$.

(a) Construct a map $\mathsf{cofib}_{(h,H)} : \mathsf{cofib}_g \to \mathsf{cofib}_f$.
(b) Use **??** to show that $\mathsf{cofib}_{\mathsf{cofib}(h,H)} \simeq \mathsf{cofib}_h$.

22.10 Use Exercise 18.4 to show that for $n \geq 0$, X is an n-type if and only if the map

$$\lambda x.\,\mathsf{const}_x : X \to (\mathbf{S}^{n+1} \to X)$$

is an equivalence.

22.11 (a) Construct for every $f : X \to Y$ a function

$$\Sigma f : \Sigma X \to \Sigma Y.$$

(b) Show that if $f \sim g$, then $\Sigma f \sim \Sigma g$.
(c) Show that $\Sigma \mathsf{id}_X \sim \mathsf{id}_{\Sigma X}$
(d) Show that

$$\Sigma(g \circ f) \sim (\Sigma g) \circ (\Sigma f).$$

for any $f : X \to Y$ and $g : Y \to Z$.

22.12 (a) Let I be a type, and let A be a family over I. Construct an equivalence

$$\Big(\bigvee_{(i:I)} \Sigma A_i\Big) \simeq \Sigma\Big(\bigvee_{(i:I)} A_i\Big).$$

(b) Show that for any type X there is an equivalence

$$\Big(\bigvee_{(x:X)} \mathbf{2}\Big) \simeq X + \mathbf{1}.$$

(c) Construct an equivalence

$$\Sigma(\mathsf{Fin}(n+1)) \simeq \bigvee_{(i:\mathsf{Fin}(n))} \mathbf{S}^1.$$

22.13 Show that $\mathsf{Fin}(n+1) * \mathsf{Fin}(m+1) \simeq \bigvee_{(i:\mathsf{Fin}(n \cdot m))} \mathbf{S}^1$, for any $n, m : \mathbb{N}$.

22.14 For any pointed set X, show that the squares

$$\begin{tikzcd} \mathbf{S}^1 \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \bigvee_{(x:X)} \mathbf{S}^1 \arrow[r] & \Sigma X \end{tikzcd} \qquad \text{and} \qquad \begin{tikzcd} X \times \mathbf{S}^1 \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \bigvee_{(x:X)} \mathbf{S}^1 \arrow[r] & \Sigma X \end{tikzcd}$$

are pushout squares.

22.15 Show that the square

$$\begin{tikzcd} \mathbf{S}^1 \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \mathbf{S}^1 \times \mathbf{S}^1 \arrow[r] & \mathbf{S}^2 \vee \mathbf{S}^1 \end{tikzcd}$$

is a pushout square.

22.16 For any type X, show that the mapping cone of the fold map $X + X \to X$ is the suspension of $X + \mathbf{1}$, i.e. show that the following square

$$\begin{tikzcd} X + X \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ X \arrow[r] & \Sigma X + \mathbf{1} \end{tikzcd}$$

is a pushout square.

22.17 Consider a map $f : A \to B$. Show that f is a k-truncated map if and only if the square

$$\begin{tikzcd} A \arrow[r, "\delta"] \arrow[d, "f"'] & A^{\mathbf{S}^{k+1}} \arrow[d, "f^{\mathbf{S}^{k+1}}"] \\ B \arrow[r, "\delta"'] & B^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square.

22.18 Show that a type A is a proposition if and only if the map $\mathsf{inl} : A \to A * A$ is an equivalence.

22.19 Let A be a type, and let P be a proposition.

(a) Show that $\mathsf{inl} : P \to P * A$ is an embedding.
(b) Show that $\mathsf{inl} : P \to P * A$ is an equivalence if and only if $\|A\| \to P$ holds.

23 Cubical diagrams

In order to proceed with the development of pullbacks and pushouts, it is useful to study commuting diagrams of the form

$$\begin{tikzcd} & C' \arrow[dl] \arrow[d] \arrow[dr] & \\ A' \arrow[d] \arrow[dr] & C \arrow[dl] \arrow[dr] & B' \arrow[dl] \arrow[d] \\ A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\ & X. & \end{tikzcd}$$

In these diagrams there are six homotopies witnessing that the faces of the cube commute, as well as a homotopy of homotopies witnessing that the cube as a whole commutes.

Once the basic definitions of cubes are established, we focus on pullbacks and pushouts that appear in different configurations in these cubical diagrams. For example, if all the vertical maps in a commuting cube are equivalences, then the top square is a pullback square if and only if the bottom square is a pullback square. In §24 we will use cubical diagrams in our formulation of the universality and descent theorems for pushouts.

In the first main theorem of this section we show that given a commuting cube in which the bottom square is a pullback square, the top square is a pullback square if and only if the induced square of fibers of the vertical maps is a pullback square. This theorem should be compared to Theorem 21.5.3, where we showed that a square is a pullback square if and only if it induces equivalences on the fibers of the vertical maps.

In our second main theorem we use the previous result to derive the 3-by-3 properties for pullbacks and pushouts.

23.1 Commuting cubes

Definition 23.1.1. A **commuting cube**

$$
\begin{tikzcd}
 & C' \arrow[dl, "p'"'] \arrow[d, "h_C"] \arrow[dr, "q'"] & \\
A' \arrow[d, "h_A"'] \arrow[dr, "f'"' near start] & C \arrow[dl, "p"' near start, crossing over] \arrow[dr, "q" near start, crossing over] & B' \arrow[d, "h_B"] \arrow[dl, "g'" near start] \\
A \arrow[dr, "f"'] & X' \arrow[d, "h_X"] & B \arrow[dl, "g"] \\
 & X &
\end{tikzcd}
$$

consists of types and maps as indicated in the diagram, equipped with

(i) homotopies

$$
\begin{aligned}
\mathsf{top} &: f' \circ p' \sim g' \circ q' \\
\mathsf{back\text{-}left} &: p \circ h_C \sim h_A \circ p' \\
\mathsf{back\text{-}right} &: q \circ h_C \sim h_B \circ q' \\
\mathsf{front\text{-}left} &: f \circ h_A \sim h_X \circ f' \\
\mathsf{front\text{-}right} &: g \circ h_B \sim h_X \circ g' \\
\mathsf{bottom} &: f \circ p \sim g \circ q
\end{aligned}
$$

witnessing that the 6 faces of the cube commute,

(ii) and a homotopy

$$
\begin{aligned}
\mathsf{coh\text{-}cube} : ((f \cdot \mathsf{back\text{-}left}) \cdot (\mathsf{front\text{-}left} \cdot p')) \cdot (h_X \cdot \mathsf{top}) \\
\sim (\mathsf{bottom} \cdot h_C) \cdot ((g \cdot \mathsf{back\text{-}right}) \cdot (\mathsf{front\text{-}right} \cdot q'))
\end{aligned}
$$

filling the cube.

In the following lemma we show that if a cube commutes, then so do its rotations and mirror symmetries (that preserve the directions of the arrows).[2] This fact is obviously true, but there is some 'path algebra' involved that we wish to demonstrate at least once.

[2]The group acting on commuting cubes of maps is the *dihedral group* D_3 which has order 6.

Lemma 23.1.2. *Consider a commuting cube*

$$\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & C \arrow[dl] \arrow[dr] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\
& X. &
\end{tikzcd}$$

Then the cubes

$$\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
C \arrow[d] \arrow[dr] & B' \arrow[dl] \arrow[dr] & A' \arrow[dl] \arrow[d] \\
B \arrow[dr] & A \arrow[d] & X' \arrow[dl] \\
& X &
\end{tikzcd}
\qquad
\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
B' \arrow[d] \arrow[dr] & A' \arrow[dl] \arrow[dr] & C \arrow[dl] \arrow[d] \\
X' \arrow[dr] & B \arrow[d] & A \arrow[dl] \\
& X &
\end{tikzcd}$$

$$\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
C \arrow[d] \arrow[dr] & A' \arrow[dl] \arrow[dr] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & B \arrow[d] & X' \arrow[dl] \\
& X. &
\end{tikzcd}
\qquad
\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & B' \arrow[dl] \arrow[dr] & C \arrow[dl] \arrow[d] \\
X' \arrow[dr] & A \arrow[d] & B \arrow[dl] \\
& X. &
\end{tikzcd}
\qquad
\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
B' \arrow[d] \arrow[dr] & C \arrow[dl] \arrow[dr] & A' \arrow[dl] \arrow[d] \\
B \arrow[dr] & X' \arrow[d] & A \arrow[dl] \\
& X. &
\end{tikzcd}$$

also commute.

Proof. We only show that the first cube commutes, which is obtained by a counter-clockwise rotation of the original cube around the axis through C' and X. The other cases are similar, and they are formalized in the accompagnying Agda library.

First we list the homotopies witnessing that the faces of the cube commute:

$$\begin{aligned}
\mathsf{top}' &:\equiv \mathsf{back\text{-}left} \\
\mathsf{back\text{-}left}' &:\equiv \mathsf{back\text{-}right}^{-1} \\
\mathsf{back\text{-}right}' &:\equiv \mathsf{top}^{-1} \\
\mathsf{front\text{-}left}' &:\equiv \mathsf{bottom}^{-1}
\end{aligned}$$

$$
\begin{aligned}
\mathsf{front\text{-}right}' &:\equiv \mathsf{front\text{-}left}^{-1} \\
\mathsf{bottom}' &:\equiv \mathsf{front\text{-}right}.
\end{aligned}
$$

Thus, to show that the cube commutes, we have to show that there is a homotopy of type

$$
\begin{aligned}
&\Big((g \cdot \mathsf{back\text{-}right}^{-1}) \cdot (\mathsf{bottom}^{-1} \cdot h_C)\Big) \cdot (f \cdot \mathsf{back\text{-}left}) \\
&\qquad \sim (\mathsf{front\text{-}right} \cdot q') \cdot \Big((h_X \cdot \mathsf{top}^{-1}) \cdot (\mathsf{front\text{-}left}^{-1} \cdot p')\Big).
\end{aligned}
$$

Recall that $h \cdot H^{-1} \sim (h \cdot H)^{-1}$ and $H^{-1} \cdot h \sim (H \cdot h)^{-1}$, so it suffices to construct a homotopy

$$
\begin{aligned}
&\Big((g \cdot \mathsf{back\text{-}right})^{-1} \cdot (\mathsf{bottom} \cdot h_C)^{-1}\Big) \cdot (f \cdot \mathsf{back\text{-}left}) \\
&\qquad \sim (\mathsf{front\text{-}right} \cdot q') \cdot \Big((h_X \cdot \mathsf{top})^{-1} \cdot (\mathsf{front\text{-}left} \cdot p')^{-1}\Big).
\end{aligned}
$$

Now we note that pointwise, our goal is of the form

$$(\varepsilon^{-1} \cdot \delta^{-1}) \cdot \alpha = \zeta \cdot (\gamma^{-1} \cdot \beta^{-1}),$$

whereas the assumption that the original cube commutes yields an identification of the form

$$(\alpha \cdot \beta) \cdot \gamma = \delta \cdot (\varepsilon \cdot \zeta)$$

Indeed, in the case that α, β, γ, δ, ε, and ζ are general identifications, we can conclude our goal using path induction on all of them. □

Lemma 23.1.3. *Given a commuting cube as in Definition 23.1.1 we obtain a commuting square*

$$
\begin{tikzcd}
\mathsf{fib}_{f_{1\check{1}1}}(x) \arrow[r] \arrow[d] & \mathsf{fib}_{f_{0\check{1}1}}(f_{\check{1}01}(x)) \arrow[d] \\
\mathsf{fib}_{f_{1\check{1}0}}(f_{10\check{1}}(x)) \arrow[r] & \mathsf{fib}_{f_{0\check{1}0}}(f_{00\check{1}}(x))
\end{tikzcd}
$$

for any $x : A_{101}$.

Lemma 23.1.4. *Consider a commuting cube*

$$
\begin{tikzcd}
& C' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & C \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\
& X, &
\end{tikzcd}
$$

If the bottom and front right squares are pullback squares, then the back left square is a pullback if and only if the top square is.

Remark 23.1.5. By rotating the cube we also obtain:

(i) If the bottom and front left squares are pullback squares, then the back right square is a pullback if and only if the top square is.

(ii) If the front left and front right squares are pullback, then the back left square is a pullback if and only if the back right square is.

By combining these statements it also follows that if the front left, front right, and bottom squares are pullback squares, then if any of the remaining three squares are pullback squares, all of them are. Cubes that consist entirely of pullback squares are sometimes called **strongly cartesian**.

23.2 Families of pullbacks

Lemma 23.2.1. *Consider a pullback square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ h$. Furthermore, consider type families P_X, P_A, P_B, and P_C over X, A, B, and C respectively, equipped with families of maps

$$\begin{aligned} f' &: \prod_{(a:A)} P_A(a) \to P_X(f(a)) \\ g' &: \prod_{(b:B)} P_B(b) \to P_X(g(b)) \\ p' &: \prod_{(c:C)} P_C(c) \to P_A(p(c)) \\ q' &: \prod_{(c:C)} P_C(c) \to P_B(q(c)), \end{aligned}$$

and for each $c : C$ a homotopy H'_c witnessing that the square

$$\begin{tikzcd} P_C(c) \arrow[rr, "q'_c"] \arrow[d, "p'_c"'] & & P_B(q(c)) \arrow[d, "g'_{q(c)}"] \\ P_A(p(c)) \arrow[r, "f'_{p(c)}"'] & P_X(f(p(c))) \arrow[r, "\mathsf{tr}_{P_X}(H(c))"'] & P_X(g(q(c))) \end{tikzcd} \tag{23.1}$$

commutes. Then the following are equivalent:

(i) For each $c : C$ the square in Eq. (23.1) is a pullback square.

(ii) The square

$$\begin{tikzcd} \sum_{(c:C)} P_C(c) \arrow[r, "\mathsf{tot}_q(q')"] \arrow[d, "\mathsf{tot}_p(p')"'] & \sum_{(b:B)} P_B(b) \arrow[d, "\mathsf{tot}_g(g')"] \\ \sum_{(a:A)} P_A(a) \arrow[r, "\mathsf{tot}_f(f')"'] & \sum_{(x:X)} P_X(x) \end{tikzcd} \tag{23.2}$$

is a pullback square.

Corollary 23.2.2. *Consider a pullback square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X, \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$, and let $c_1, c_2 : C$. Then the square

$$\begin{tikzcd} (c_1 = c_2) \arrow[rr, "\mathsf{ap}_q"] \arrow[d, "\mathsf{ap}_p"'] & & (q(c_1) = q(c_2)) \arrow[d, "\lambda\beta.\, H(c_1) \cdot \mathsf{ap}_g(\beta)"] \\ (p(c_1) = p(c_2)) \arrow[rr, "\lambda\alpha.\, \mathsf{ap}_f(\alpha) \cdot H(c_2)"'] & & f(p(c_1)) = g(q(c_2)), \end{tikzcd}$$

commutes and is a pullback square.

Theorem 23.2.3. *Consider a commuting cube*

$$\begin{tikzcd} & C' \arrow[dl] \arrow[d] \arrow[dr] & \\ A' \arrow[d] \arrow[dr] & C \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\ A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\ & X & \end{tikzcd}$$

in which the bottom square is a pullback square. Then the following are equivalent:

(i) The top square is a pullback square.

(ii) The square

$$\begin{tikzcd} \mathsf{fib}_\gamma(c) \arrow[r] \arrow[d] & \mathsf{fib}_\beta(q(c)) \arrow[d] \\ \mathsf{fib}_\alpha(p(c)) \arrow[r] & \mathsf{fib}_\varphi(f(p(c))) \end{tikzcd}$$

is a pullback square for each $c : C$.

23.3 The 3-by-3-properties for pullbacks and pushouts

Theorem 23.3.1. *Consider a commuting diagram of the form*

$$\begin{tikzcd}
AA \arrow[r, "Af"] \arrow[d, "fA"'] \arrow[dr, phantom, "\Rightarrow"] & AX \arrow[d, "fX"'] \arrow[dr, phantom, "\Leftarrow"] & AB \arrow[l, "Ag"'] \arrow[d, "gB"] \\
XA \arrow[r, "Xf"] & XX & XB \arrow[l, "Xg"'] \\
BA \arrow[r, "Bf"'] \arrow[u, "gA"] & BX \arrow[u, "gX"] & BB \arrow[l, "Bg"] \arrow[u, "gB"']
\end{tikzcd}$$

with homotopies

$$ff : Xf \circ fA \sim Af \circ fX$$
$$fg : Xg \circ gB \sim Ag \circ fX$$
$$gf :$$

filling the (small) squares. Furthermore, consider pullback squares

$$\begin{tikzcd}
AC \arrow[r] \arrow[d] & AB \arrow[d] \\
AA \arrow[r] & AX
\end{tikzcd}
\qquad
\begin{tikzcd}
XC \arrow[r] \arrow[d] & XB \arrow[d] \\
XA \arrow[r] & XX
\end{tikzcd}
\qquad
\begin{tikzcd}
BC \arrow[r] \arrow[d] & BB \arrow[d] \\
BA \arrow[r] & BX
\end{tikzcd}$$

$$\begin{tikzcd}
CA \arrow[r] \arrow[d] & BA \arrow[d] \\
AA \arrow[r] & XA
\end{tikzcd}
\qquad
\begin{tikzcd}
CX \arrow[r] \arrow[d] & BX \arrow[d] \\
AX \arrow[r] & XX
\end{tikzcd}
\qquad
\begin{tikzcd}
CB \arrow[r] \arrow[d] & BB \arrow[d] \\
AB \arrow[r] & XB.
\end{tikzcd}$$

Finally, consider a commuting square

$$\begin{tikzcd}
D_3 \arrow[r] \arrow[d] & D_2 \arrow[d] \\
D_0 \arrow[r] & D_1.
\end{tikzcd}$$

Then the following are equivalent:

(i) *This square is a pullback square.*

(ii) *The induced square*

$$\begin{tikzcd}
D_3 \arrow[r] \arrow[d] & C_3 \arrow[d] \\
A_3 \arrow[r] & B_3
\end{tikzcd}$$

is a pullback square.

Proof. First we construct an equivalence

$$(A_0 \times_{B_0} C_0) \times_{(A_1 \times_{B_1} C_1)} (A_2 \times_{B_2} C_2) \simeq (A_0 \times_{A_1} A_2) \times_{(B_0 \times_{B_1} B_2)} (C_0 \times_{C_1} C_2).$$

Now it follows that we have an equivalence

$$\mathsf{cone}(f_0, g_0)$$

□

Exercises

23.1 Some exercises.

24 Universality and descent for pushouts

We begin this section with the idea that pushouts can be presented as higher inductive types. The general idea behind higher inductive types is that we can introduce new inductive types not only with constructors at the level of points, but also with constructors at the level of identifications. Pushouts form a basic class of examples that can be obtained as higher inductive types, because they come equipped with the structure of a cocone. The cocone (i, j, H) in the commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & C \end{tikzcd}$$

equips the type C with two *point constructors*

$$i : A \to C$$
$$j : B \to C$$

and a *path constructor*

$$H : \prod_{(s:S)} i(f(s)) = j(g(s))$$

that provides an identification $H(s) : i(f(s)) = j(g(s))$ for every $s : S$. The induction principle then specifies how to construct sections of families over C. Naturally, it takes not only the point constructors i and j, but also the path constructor H into account.

The induction principle is one of several equivalent characterizations of pushouts. We will prove a theorem providing five equivalent characterizations of homotopy pushouts. Two of those we have already seen in Theorem 22.3.2: the universal property and the pullback property. The other three are

(i) the *dependent pullback property*,

(ii) the *dependent universal property*,

(iii) the *induction principle*.

An implication that is particularly useful among our five characterizations of pushouts, is the fact that the pullback property implies the dependent pullback property. We use the dependent pullback property to derive the *universality of pushouts* (not to be confused with the universal property of pushouts), showing that for any commuting cube

$$\begin{tikzcd}
 & S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & D' \arrow[d] & B \arrow[dl] \\
 & D &
\end{tikzcd}$$

in which the back left and right squares are pullback squares, if the front left and right squares are also pullback squares, then so is the induced square

$$\begin{tikzcd}
A' \sqcup^{S'} B' \arrow[r, dotted] \arrow[d, dotted] & D' \arrow[d] \\
A \sqcup^{S} B \arrow[r, dotted] & D
\end{tikzcd}$$

We then observe that the univalence axiom can be used together with the universal property of pushouts to obtain such families over pushouts in the first place. We prove the descent theorem, which asserts that for any diagram of the form

$$\begin{tikzcd}
 & S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] & S \arrow[dl] \arrow[dr] & B' \arrow[d] \\
A \arrow[dr] & & B \arrow[dl] \\
 & C &
\end{tikzcd}$$

in which the bottom square is a pushout square and the back left and right squares are pullback squares, there is a unique way of extending this to a commuting cube

$$\begin{tikzcd}
 & S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr, dotted] & S \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl, dotted] \arrow[d] \\
A \arrow[dr] & C' \arrow[d, dotted] & B \arrow[dl] \\
 & C &
\end{tikzcd}$$

in which also the front left and right squares are pullback squares. Thus the converse of the universality theorem for pushouts also follows. The descent property used to show that pullbacks distribute over pushouts, and to compute the fibers of maps out of pushouts (the source of many exercises).

We note that the computation rules in our treatment for the induction principle of homotopy pushouts are weak. In other words, they are identifications. In this course we have no need for judgmental computation rules. Our focus is instead on universal properties. We refer the reader who is interested in the more 'traditional' higher inductive types with judgmental computation rules to [3].

24.1 Five equivalent characterizations of homotopy pushouts

Consider a commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & H \end{tikzcd} \tag{24.1}$$

with $H : i \circ f \sim j \circ g$, where we will sometimes write $\mathcal{S}$ for the span $A \leftarrow S \rightarrow B$. Our first goal is to formulate the induction principle for pushouts, which specifies how to construct a section of an arbitrary type family P over X. Like the induction principle for the circle, the induction principle of pushouts has to take both the point constructors and the path constructors of X into account. In our case, the point constructors are the maps

$$\begin{aligned} i &: A \to X \\ j &: B \to X, \end{aligned}$$

and the path constructor is the homotopy

$$H : \prod_{(s:S)} i(f(s)) = j(g(s)).$$

Therefore, we obtain for any section $h : \prod_{(x:X)} P(x)$ a triple (h_A, h_B, h_S) consisting of

$$\begin{aligned} h_A &: \prod_{(a:A)} P(i(a)) \\ h_B &: \prod_{(b:B)} P(j(b)) \\ h_S &: \prod_{(s:S)} \mathsf{tr}_P(H(s), h(i(f(s)))) = h(j(g(s))). \end{aligned}$$

The dependent functions h_A and h_B are simply given by

$$\begin{aligned} h_A &:\equiv h \circ i \\ h_B &:\equiv h \circ j. \end{aligned}$$

The homotopy h_S is defined by $h_S(s) :\equiv \mathsf{apd}_h(H(s))$, using the dependent action on paths of h. We call such triples (h_A, h_B, h_S) **dependent cocones** on P over the cocone (i, j, H), and will write $\mathsf{dep\text{-}cocone}_{(i,j,H)}(P)$ for this type of dependent cocones. Thus, we have a function

$$\mathsf{ev\text{-}pushout}(P) : \left(\prod_{(x:X)} P(x)\right) \to \mathsf{dep\text{-}cocone}_{(i,j,H)}(P).$$

We are now in position to define the induction principle and the dependent universal property of pushouts.

Definition 24.1.1. We say that X satisfies the **induction principle of the pushout of** $\mathcal{S}$ if the function

$$\mathsf{ev\text{-}pushout}(P) : \Big(\prod_{(x:X)} P(x)\Big) \to \mathsf{dep\text{-}cocone}_{(i,j,H)}(P).$$

has a section for every type family P over X.

Definition 24.1.2. We say that X satisfies the **dependent universal property of the pushout of** $\mathcal{S}$ if the function

$$\mathsf{ev\text{-}pushout}(P) : \Big(\prod_{(x:X)} P(x)\Big) \to \mathsf{dep\text{-}cocone}_{(i,j,H)}(P).$$

is an equivalence for every type family P over X.

Remark 24.1.3. For (h_A, h_B, h_S) and (h'_A, h'_B, h'_S) in $\mathsf{dep\text{-}cocone}_{(i,j,H)}(P)$, the type of identifications $(h_A, h_B, h_S) = (h'_A, h'_B, h'_S)$ is equivalent to the type of triples (K_A, K_B, K_S) consisting of

$$\begin{aligned} K_A &: \prod_{(a:A)} h_A(a) = h'_A(a) \\ K_B &: \prod_{(b:B)} h_B(b) = h'_B(b), \end{aligned}$$

and a homotopy K_S witnessing that the square

$$\begin{tikzcd} \mathsf{tr}_P(H(s), h_A(f(s))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(H(s))}(K_A(f(s)))"] \arrow[d, equal, "h_S(s)"'] & \mathsf{tr}_P(H(s), h'_A(f(s))) \arrow[d, equal, "h'_S(s)"] \\ h_B(g(s)) \arrow[r, equal, "K_B(g(s))"'] & h_B(g(s)) \end{tikzcd}$$

commutes for every $s : S$.

Therefore we see that the induction principle of the pushout of $\mathcal{S}$ provides us, for every dependent cocone (h_A, h_B, h_S) of P over (i, j, H), with a dependent function $h : \prod_{(x:A)} P(x)$ equipped with homotopies

$$\begin{aligned} K_A &: \prod_{(a:A)} h(i(a)) = h_A(a) \\ K_B &: \prod_{(b:B)} h(j(b)) = h_B(b), \end{aligned}$$

and a homotopy K_S witnessing that the square

$$\begin{tikzcd} \mathsf{tr}_P(H(s), h(i(f(s)))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(H(s))}(K_A(f(s)))"] \arrow[d, equal, "\mathsf{apd}_h(H(s))"'] & \mathsf{tr}_P(H(s), h_A(f(s))) \arrow[d, equal, "h_S(s)"] \\ h(j(g(s))) \arrow[r, equal, "K_B(g(s))"'] & h_B(g(s)) \end{tikzcd}$$

commutes for every $s : S$. These homotopies are the **computation rules** for pushouts. The dependent universal property is equivalent to the assertion that for every dependent cocone (h_A, h_B, h_S), the type of quadruples (h, K_A, K_B, K_S) is contractible.

Theorem 24.1.4. *Consider a commuting square*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & C \end{tikzcd} \tag{24.2}$$

with $H : (i \circ f) \sim (j \circ g)$. Then the following are equivalent:

(i) The square in Eq. (24.2) is a pushout square.

(ii) The square in Eq. (24.2) satisfies the pullback property of pushouts.

*(iii) The square satisfies the **dependent pullback property** of pushouts: For every family P over C, the square*

$$\begin{tikzcd} \prod_{(z:C)} P(z) \arrow[rr, "h \mapsto h \circ j"] \arrow[d, "h \mapsto h \circ i"'] & & \prod_{(y:B)} P(j(y)) \arrow[d, "h \mapsto h \circ g"] \\ \prod_{(x:A)} P(i(x)) \arrow[r, "h \mapsto h \circ f"'] & \prod_{(s:S)} P(i(f(s))) \arrow[r, "\lambda h.\, \lambda s.\, \mathsf{tr}_P(H(s), h(s))"'] & \prod_{(s:S)} P(j(g(s))), \end{tikzcd} \tag{24.3}$$

which commutes by the homotopy

$$\lambda h.\, \mathsf{eq\text{-}htpy}(\lambda s.\, \mathsf{apd}_h(H(s))),$$

is a pullback square.

*(iv) The type C satisfies the **dependent universal property** of pushouts.*

*(v) The type C satisfies the **induction principle** of pushouts.*

Proof. We have already seen in Theorem 22.3.2 that (i) and (ii) are equivalent.

To see that (ii) implies (iii), note that we have a commuting cube

$$\begin{tikzcd}[column sep=small] & & \sum_{(h:C \to C)} \prod_{(c:C)} P(h(c)) \arrow[dll] \arrow[d] \arrow[drr] & & \\ \sum_{(h:A \to C)} \prod_{(a:A)} P(h(a)) \arrow[d] \arrow[drr] & & \big(\sum_{(c:C)} P(c)\big)^C \arrow[dll] \arrow[drr] & & \sum_{(h:B \to C)} \prod_{(b:B)} P(h(b)) \arrow[dll] \arrow[d] \\ \big(\sum_{(c:C)} P(c)\big)^A \arrow[drr] & & \sum_{(h:S \to C)} \prod_{(s:S)} P(h(s)) \arrow[d] & & \big(\sum_{(c:C)} P(c)\big)^B \arrow[dll] \\ & & \big(\sum_{(c:C)} P(c)\big)^S & & \end{tikzcd}$$

in which the vertical maps are equivalences. Moreover, the bottom square is a pullback square by the pullback property of pushouts, so we conclude that the top square is a pullback square.

Since this is a square of total spaces over a pullback square, we invoke Lemma 23.2.1 to conclude that for each $h : C \to C$, the square

$$\begin{tikzcd}
\prod_{(c:C)} P(h(c)) \arrow[rr] \arrow[d] & & \prod_{(b:B)} P(h(j(b))) \arrow[d] \\
\prod_{(a:A)} P(h(i(a))) \arrow[r] & \prod_{(s:S)} P(h(i(f(s)))) \arrow[r, "\mathsf{tr}_{((k:S\to C)\mapsto \prod_{(s:S)} P(k(s)))}(\mathsf{eq\text{-}htpy}(h \cdot H))"'] & \prod_{(s:S)} P(h(j(g(s))))
\end{tikzcd}$$

is a pullback square. Note that the transport with respect to the family $k \mapsto \prod_{(s:S)}(Pk(s))$ along the identification $\mathsf{eq\text{-}htpy}(h \cdot H)$ is homotopic to the map

$$\lambda h.\, \lambda s.\, \mathsf{tr}_{P \circ h}(H(s), h(s)) : \prod_{(s:S)} P(h(i(f(s)))) \to \prod_{(s:S)} P(h(j(g(s)))).$$

Therefore we conclude that the square

$$\begin{tikzcd}
\prod_{(c:C)} P(h(c)) \arrow[rr] \arrow[d] & & \prod_{(b:B)} P(h(j(b))) \arrow[d] \\
\prod_{(a:A)} P(h(i(a))) \arrow[r] & \prod_{(s:S)} P(h(i(f(s)))) \arrow[r, "\lambda h.\, \lambda s.\, \mathsf{tr}_{P\circ h}(H(s), h(s))"'] & \prod_{(s:S)} P(h(j(g(s))))
\end{tikzcd}$$

is a pullback square for each $h : C \to C$. Using the case $h \equiv \mathsf{id} : C \to C$ we conclude that the cocone (i, j, H) satisfies the dependent pullback property.

To see that (iii) implies (ii) we recall that transport with respect to a trivial family is homotopic to the identity function. Thus we obtain the pullback property from the dependent pullback property using the trivial family $\lambda c.\, T$ over C.

To see that (iii) implies (iv) we note that $\mathsf{ev\text{-}pushout}(P)$ is an equivalence if and only if the gap map of the square in Eq. (24.3) is an equivalence.

It is clear that (iv) implies (v), so it remains to show that (v) implies (iv). If X satisfies the induction principle of pushouts, then the map

$$\mathsf{ev\text{-}pushout} : \Big(\prod_{(x:X)} P(x)\Big) \to \mathsf{dep\text{-}cocone}_{(i,j,H)}(P)$$

has a section, i.e., it comes equipped with

$$\mathsf{ind\text{-}pushout} : \mathsf{dep\text{-}cocone}_{(i,j,H)}(P) \to \Big(\prod_{(x:X)} P(x)\Big)$$

$$\mathsf{comp\text{-}pushout} : \mathsf{ev\text{-}pushout} \circ \mathsf{ind\text{-}pushout} \sim \mathsf{id}.$$

To see that ev-pushout is an equivalence it therefore suffices to construct a homotopy

$$\mathsf{ind\text{-}pushout}(\mathsf{ev\text{-}pushout}(h)) \sim h$$

for any $h : \prod_{(x:X)} P(x)$. From the fact that ind-pushout is a section of ev-pushout we obtain an identification

$$\mathsf{ev\text{-}pushout}(\mathsf{ind\text{-}pushout}(\mathsf{ev\text{-}pushout}(h))) = \mathsf{ev\text{-}pushout}(h).$$

Therefore we observe that it suffices to construct a homotopy $h \sim h'$ for any two functions $h, h' : \prod_{(x:X)} P(x)$ that come equipped with an identification

$$\mathsf{ev\text{-}pushout}(h) = \mathsf{ev\text{-}pushout}(h').$$

Now we recall from Remark 24.1.3 that this type of identifications is equivalent to the type of triples (K_A, K_B, K_S) consisting of

$$K_A : \prod_{(a:A)} h(i(a)) = h'(i(a))$$
$$K_B : \prod_{(b:B)} h(j(b)) = h'(j(b))$$

and a homotopy K_S witnessing that the square

$$\begin{tikzcd}
\mathsf{tr}_P(H(s), h(i(f(s)))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(H(s))}(K_A(f(s)))"] \arrow[d, equal, "\mathsf{apd}_h(H(s))"'] & \mathsf{tr}_P(H(s), h'(i(f(s)))) \arrow[d, equal, "\mathsf{apd}_{h'}(H(s))"] \\
h(j(g(s))) \arrow[r, equal, "K_B(g(s))"'] & h'(j(g(s)))
\end{tikzcd}$$

commutes for every $s : S$. Note that from such an identification $K_S(s)$ we also obtain an identification

$$\mathsf{K}'_\mathsf{S}(\mathsf{s}) : \mathsf{tr}_{x \mapsto h(x) = h'(x)}(H(s), K_A(f(s))) = K_B(g(s)).$$

Indeed, by path inducgtion on $p : x = x'$ we obtain an identification $\mathsf{tr}_{x \mapsto h(x) = h'(x)}(p, q) = q'$, for any $p : x = x'$, any $q : h(x) = h'(x)$ and any $q' : h(x') = h'(x')$ for which the square

$$\begin{tikzcd}
\mathsf{tr}_P(p, h(x)) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(p)}(q)"] \arrow[d, equal, "\mathsf{apd}_h(p)"'] & \mathsf{tr}_P(p, h'(x)) \arrow[d, equal, "\mathsf{apd}_{h'}(p)"] \\
h(x') \arrow[r, equal, "q'"'] & h'(x')
\end{tikzcd}$$

Now we see that the triple (K_A, K_B, K'_S) forms a dependent cocone on the family $x \mapsto h(x) = h'(x)$. Therefore we obtain a homotopy $h \sim h'$ as an application of the induction principle for pushouts at the family $x \mapsto h(x) = h'(x)$. □

24.2 Type families over pushouts

Given a pushout square

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\
A \arrow[r, "i"'] & X.
\end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, and a family $P : X \to \mathcal{U}$, we obtain

$$P \circ i : A \to \mathcal{U}$$
$$P \circ j : B \to \mathcal{U}$$
$$\lambda x.\, \mathsf{tr}_P(H(x)) : \prod_{(x:S)} P(i(f(x))) \simeq P(j(g(x))).$$

Our goal in the current section is to show that the triple (P_A, P_B, P_S) consisting of $P_A :\equiv P \circ i$, $P_B :\equiv P \circ j$, and $P_S :\equiv \lambda x.\, \mathsf{tr}_P(H(x))$ characterizes the family P over X.

Definition 24.2.1. Consider a commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, where all types involved are in $\mathcal{U}$. The type $\mathsf{Desc}(\mathcal{S})$ of **descent data** for X, is defined defined to be the type of triples (P_A, P_B, P_S) consisting of

$$\begin{aligned} P_A &: A \to \mathcal{U} \\ P_B &: B \to \mathcal{U} \\ P_S &: \textstyle\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x)). \end{aligned}$$

Definition 24.2.2. Given a commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, we define the map

$$\mathsf{desc\text{-}fam}_{\mathcal{S}}(i, j, H) : (X \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{S})$$

by $P \mapsto (P \circ i, P \circ j, \lambda x.\, \mathsf{tr}_P(H(x)))$.

Theorem 24.2.3. *Consider a pushout square*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, where all types involved are in $\mathcal{U}$, and suppose we have

$$\begin{aligned} P_A &: A \to \mathcal{U} \\ P_B &: B \to \mathcal{U} \\ P_S &: \textstyle\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x)). \end{aligned}$$

Then the function

$$\mathsf{desc\text{-}fam}_{\mathcal{S}}(i, j, H) : (X \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{S})$$

is an equivalence.

Proof. By the 3-for-2 property of equivalences it suffices to construct an equivalence $\varphi : \mathsf{cocone}_{\mathcal{S}}(\mathcal{U}) \to \mathsf{Desc}(\mathcal{S})$ such that the triangle

$$\begin{tikzcd} & \mathcal{U}^X \arrow[dl, "\mathsf{cocone\text{-}map}_{\mathcal{S}}(i,j,H)"'] \arrow[dr, "\mathsf{desc\text{-}fam}_{\mathcal{S}}(i,j,H)"] & \\ \mathsf{cocone}_{\mathcal{S}}(\mathcal{U}) \arrow[rr, dotted, "\simeq", "\varphi"'] & & \mathsf{Desc}(\mathcal{S}) \end{tikzcd}$$

commutes.

Since we have equivalences

$$\mathsf{equiv\text{-}eq} : \Big(P_A(f(x)) = P_B(g(x))\Big) \simeq \Big(P_A(f(x)) \simeq P_B(g(x))\Big)$$

for all $x : S$, we obtain by Exercise 11.12 an equivalence on the dependent products

$$\Big(\prod_{(x:S)} P_A(f(x)) = P_B(g(x))\Big) \to \Big(\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x))\Big).$$

We define φ to be the induced map on total spaces. Explicitly, we have

$$\varphi :\equiv \lambda(P_A, P_B, K).\, (P_A, P_B, \lambda x.\, \mathsf{equiv\text{-}eq}(K(x))).$$

Then φ is an equivalence by Theorem 9.1.3, and the triangle commutes by **??**. □

Corollary 24.2.4. *Consider descent data (P_A, P_B, P_S) for a pushout square as in Theorem 24.2.3. Then the type of quadruples (P, e_A, e_B, e_S) consisting of a family $P : X \to \mathcal{U}$ equipped with two families of equivalences*

$$e_A : \prod_{(a:A)} P_A(a) \simeq P(i(a))$$
$$e_B : \prod_{(b:B)} P_B(a) \simeq P(j(b))$$

and a homotopy e_S witnessing that the square

$$\begin{tikzcd}
P_A(f(x)) \arrow[r, "e_A(f(x))"] \arrow[d, "P_S(x)"'] & P(i(f(x))) \arrow[d, "\mathsf{tr}_P(H(x))"] \\
P_B(g(x)) \arrow[r, "e_B(g(x))"'] & P(j(g(x)))
\end{tikzcd}$$

commutes, is contractible.

Proof. The fiber of this map at (P_A, P_B, P_S) is equivalent to the type of quadruples (P, e_A, e_B, e_S) as described in the theorem, which are contractible by Theorem 8.3.6. □

24.3 The flattening lemma for pushouts

In this section we consider a pushout square

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\
A \arrow[r, "i"'] & X.
\end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, descent data

$$P_A : A \to \mathcal{U}$$
$$P_B : B \to \mathcal{U}$$
$$P_S : \prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x)),$$

and a family $P : X \to \mathcal{U}$ equipped with

$$e_A : \prod_{(a:A)} P_A(a) \simeq P(i(a))$$
$$e_B : \prod_{(b:B)} P_B(a) \simeq P(j(b))$$

and a homotopy e_S witnessing that the square

$$\begin{tikzcd} P_A(f(x)) \arrow[r, "e_A(f(x))"] \arrow[d, "P_S(x)"'] & P(i(f(x))) \arrow[d, "\mathsf{tr}_P(H(x))"] \\ P_B(g(x)) \arrow[r, "e_B(g(x))"'] & P(j(g(x))) \end{tikzcd}$$

commutes.

Definition 24.3.1. We define a commuting square

$$\begin{tikzcd} \sum_{(x:S)} P_A(f(x)) \arrow[r, "g'"] \arrow[d, "f'"'] & \sum_{(b:B)} P_B(b) \arrow[d, "j'"] \\ \sum_{(a:A)} P_A(a) \arrow[r, "i'"'] & \sum_{(x:X)} P(x) \end{tikzcd}$$

with a homotopy $H' : i' \circ f' \sim j' \circ g'$. We will write $\mathcal{S}'$ for the span

$$\textstyle\sum_{(a:A)} P_A(a) \xleftarrow{f'} \sum_{(x:S)} P_A(f(x)) \xrightarrow{g'} \sum_{(b:B)} P_B(b).$$

Construction. We define

$$\begin{aligned} f' &:\equiv \mathsf{tot}_f(\lambda x.\, \mathsf{id}_{P_A(f(x))}) \\ g' &:\equiv \mathsf{tot}_g(e_S) \\ i' &:\equiv \mathsf{tot}_i(e_A) \\ j' &:\equiv \mathsf{tot}_j(e_B). \end{aligned}$$

Then it remains to construct a homotopy $H' : i' \circ f' \sim j' \circ g'$. In order to construct this homotopy, we have to construct an identification

$$(i(f(x)), e_A(y)) = (j(g(x)), e_B(e_S(y)))$$

for any $x : S$ and $y : P_A(f(x))$. Note that have the identification

$$\mathsf{eq\text{-}pair}(H(x), e_S(x, y)^{-1})$$

of this type. □

Lemma 24.3.2 (The flattening lemma)**.** *The commuting square*

$$\begin{tikzcd} \sum_{(x:S)} P_A(f(x)) \arrow[r, "g'"] \arrow[d, "f'"'] & \sum_{(b:B)} P_B(b) \arrow[d, "j'"] \\ \sum_{(a:A)} P_A(a) \arrow[r, "i'"'] & \sum_{(x:X)} P(x) \end{tikzcd}$$

is a pushout square.

Proof. To show that the square of total spaces satisfies the pullback property of pullbacks, note that we have a commuting cube

$$\begin{tikzcd}
 & & T^{\Sigma_{(x:X)} P(x)} \arrow[dll] \arrow[d] \arrow[drr] & & \\
T^{\Sigma_{(a:A)} P_A(a)} \arrow[d] \arrow[drr] & & \prod_{(x:X)} T^{P(x)} \arrow[dll] \arrow[drr] & & T^{\Sigma_{(b:B)} P_B(b)} \arrow[dll] \arrow[d] \\
\prod_{(a:A)} T^{P_A(a)} \arrow[drr] & & T^{\Sigma_{(x:S)} P_A(f(x))} \arrow[d] & & \prod_{(b:B)} T^{P_B(b)} \arrow[dll] \\
 & & \prod_{(x:S)} T^{P_A(f(x))} & &
\end{tikzcd}$$

for any type T. In this cube, the vertical maps are all equivalences, and the bottom square is a pullback square by the dependent pullback property of pushouts. Therefore it follows that the top square is a pullback square. □

24.4 The universality theorem

Theorem 24.4.1. *Consider two pushout squares*

$$\begin{tikzcd}
S' \arrow[r] \arrow[d] & B' \arrow[d] \\
A' \arrow[r] & C'
\end{tikzcd}
\qquad
\begin{tikzcd}
S \arrow[r] \arrow[d] & B \arrow[d] \\
A \arrow[r] & C
\end{tikzcd}$$

and a commuting cube

$$\begin{tikzcd}
 & S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl] \arrow[dr] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & D' \arrow[d] & B \arrow[dl] \\
 & D &
\end{tikzcd}$$

in which the back left and right squares are pullback squares. The following are equivalent:

(i) The front left and right squares are pullback squares.

(ii) The induced commuting square

$$\begin{tikzcd}
C' \arrow[r, dotted] \arrow[d, dotted] & D' \arrow[d] \\
C \arrow[r, dotted] & D
\end{tikzcd}$$

is a pullback square.

24.5 The descent property for pushouts

In the previous section there was a significant role for families of equivalences, and we know by Theorems 21.5.2 and 21.5.3: families of equivalences indicate the presence of pullbacks. In this section we reformulate the results of the previous section using pullbacks where we used families of equivalences before, to obtain new and useful results. We begin by considering the type of descent data from the perspective of pullback squares.

Definition 24.5.1. Consider a span $\mathcal{S}$ from A to B, and a span $\mathcal{S}'$ from A' to B'. A **cartesian transformation** of spans from $\mathcal{S}'$ to $\mathcal{S}$ is a diagram of the form

$$\begin{tikzcd} A' \arrow[d, "h_A"'] & S' \arrow[l, "f'"'] \arrow[r, "g'"] \arrow[d, "h_S"'] & B' \arrow[d, "h_B"] \\ A & S \arrow[l, "f"] \arrow[r, "g"'] & B \end{tikzcd}$$

with $F : f \circ h_S \sim h_A \circ f'$ and $G : g \circ h_S \sim h_B \circ g'$, where both squares are pullback squares.

The type $\mathsf{cart}(\mathcal{S}', \mathcal{S})$ of cartesian transformation is the type of tuples

$$(h_A, h_S, h_B, F, G, p_f, p_g)$$

where $p_f : \mathsf{is\text{-}pullback}(h_S, h_A, F)$ and $p_g : \mathsf{is\text{-}pullback}(h_S, h_B, G)$, and we write

$$\mathsf{Cart}(\mathcal{S}) :\equiv \textstyle\sum_{(A',B':\mathcal{U})}\sum_{(\mathcal{S}':\mathsf{span}(A',B'))}\mathsf{cart}(\mathcal{S}', \mathcal{S}).$$

Lemma 24.5.2. *There is an equivalence*

$$\mathsf{cart\text{-}desc}_{\mathcal{S}} : \mathsf{Desc}(\mathcal{S}) \to \mathsf{Cart}(\mathcal{S}).$$

Proof. Note that by Theorem 21.5.6 it follows that the types of triples (f', F, p_f) and (g', G, p_g) are equivalent to the types of families of equivalences

$$\textstyle\prod_{(x:S)}\mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_A}(f(x))$$
$$\textstyle\prod_{(x:S)}\mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_B}(g(x))$$

respectively. Furthermore, by **??** the types of pairs (S', h_S), (A', h_A), and (B', h_B) are equivalent to the types $S \to \mathcal{U}$, $A \to \mathcal{U}$, and $B \to \mathcal{U}$, respectively. Therefore it follows that the type $\mathsf{Cart}(\mathcal{S})$ is equivalent to the type of tuples $(Q, P_A, \varphi, P_B, P_S)$ consisting of

$$\begin{aligned} Q &: S \to \mathcal{U} \\ P_A &: A \to \mathcal{U} \\ P_B &: B \to \mathcal{U} \\ \varphi &: \textstyle\prod_{(x:S)}Q(x) \simeq P_A(f(x)) \\ P_S &: \textstyle\prod_{(x:S)}Q(x) \simeq P_B(g(x)). \end{aligned}$$

However, the type of φ is equivalent to the type $P_A \circ f = Q$. Thus we see that the type of pairs (Q, φ) is contractible, so our claim follows. $\square$

Definition 24.5.3. We define an operation

$$\mathsf{cart\text{-}map}_{\mathcal{S}} : \left(\sum_{(X':\mathcal{U})} X' \to X\right) \to \mathsf{Cart}(\mathcal{S}).$$

Construction. Let $X' : \mathcal{U}$ and $h_X : X' \to X$. Then we define the types

$$A' :\equiv A \times_X X'$$
$$B' :\equiv B \times_X X'.$$

Next, we define a span $\mathcal{S}' :\equiv (S', f', g')$ from A' to B'. We take

$$S' :\equiv S \times_A A'$$
$$f' :\equiv \pi_2.$$

To define g', let $s : S$, let $(a, x', p) : A \times_X X'$, and let $q : f(s) = a$. Our goal is to construct a term of type $B \times_X X'$. We have $g(s) : B$ and $x' : X'$, so it remains to show that $j(g(s)) = h_X(x')$. We construct such an identification as a concatenation

$$j(g(s)) \overset{H(s)^{-1}}{=\!=\!=} i(f(s)) \overset{\mathsf{ap}_i(q)}{=\!=\!=} i(a) \overset{p}{=\!=\!=} h_X(x').$$

To summarize, the map g' is defined as

$$g' :\equiv \lambda(s, (a, x', p), q).\, (g(s), x', H(s)^{-1} \cdot (\mathsf{ap}_i(q) \cdot p)).$$

Then we have commuting squares

$$\begin{tikzcd}
A \times_X X' \arrow[d] & S \times_A A' \arrow[l] \arrow[r] \arrow[d] & B \times_X X' \arrow[d] \\
A & S \arrow[l] \arrow[r] & B.
\end{tikzcd}$$

Moreover, these squares are pullback squares by Theorem 21.5.8. □

The following theorem is analogous to Theorem 24.2.3.

Theorem 24.5.4 (The descent theorem for pushouts). *The operation* $\mathsf{cart\text{-}map}_{\mathcal{S}}$ *is an equivalence*

$$\left(\sum_{(X':\mathcal{U})} X' \to X\right) \simeq \mathsf{Cart}(\mathcal{S})$$

Proof. It suffices to show that the square

$$\begin{tikzcd}
X \to \mathcal{U} \arrow[r, "\mathsf{desc\text{-}fam}_{\mathcal{S}}(i,j,H)"] \arrow[d, "\mathsf{map\text{-}fam}_X"'] & \mathsf{Desc}(\mathcal{S}) \arrow[d, "\mathsf{cart\text{-}desc}_{\mathcal{S}}"] \\
\sum_{(X':\mathcal{U})} X' \to X \arrow[r, "\mathsf{cart\text{-}map}_{\mathcal{S}}"'] & \mathsf{Cart}(\mathcal{S})
\end{tikzcd}$$

commutes. To see that this suffices, note that the operation $\mathsf{map\text{-}fam}_X$ is an equivalence by **??**, the operation $\mathsf{desc\text{-}fam}_{\mathcal{S}}(i, j, H)$ is an equivalence by Theorem 24.2.3, and the operation $\mathsf{cart\text{-}desc}_{\mathcal{S}}$ is an equivalence by Lemma 24.5.2.

To see that the square commutes, note that the composite

$$\mathsf{cart\text{-}map}_{\mathcal{S}} \circ \mathsf{map\text{-}fam}_X$$

takes a family $P : X \to \mathcal{U}$ to the cartesian transformation of spans

$$\begin{tikzcd} A \times_X \tilde{P} \arrow[d, "\pi_1"'] & S \times_A \left(A \times_X \tilde{P}\right) \arrow[l] \arrow[r] \arrow[d, "\pi_1"'] & B \times_X \tilde{P} \arrow[d, "\pi_1"] \\ A & S \arrow[l] \arrow[r] & B, \end{tikzcd}$$

where $\tilde{P} :\equiv \sum_{(x:X)} P(x)$.

The composite

$$\mathsf{cart\text{-}desc}_{\mathcal{S}} \circ \mathsf{desc\text{-}fam}_X$$

takes a family $P : X \to \mathcal{U}$ to the cartesian transformation of spans

$$\begin{tikzcd} \sum_{(a:A)} P(i(a)) \arrow[d] & \sum_{(s:S)} P(i(f(s))) \arrow[l] \arrow[r] \arrow[d] & \sum_{(b:B)} P(j(b)) \arrow[d] \\ A & S \arrow[l] \arrow[r] & B \end{tikzcd}$$

These cartesian natural transformations are equal by Lemma 21.5.1 □

Since $\mathsf{cart\text{-}map}_{\mathcal{S}}$ is an equivalence it follows that its fibers are contractible. This is essentially the content of the following corollary.

Corollary 24.5.5. *Consider a diagram of the form*

$$\begin{tikzcd} & S' \arrow[dl, "f'"'] \arrow[d, "h_S"'] \arrow[dr, "g'"] & \\ A' \arrow[d, "h_A"'] & S \arrow[dl, "f"'] \arrow[dr, "g"] & B' \arrow[d, "h_B"] \\ A \arrow[dr, "i"'] & & B \arrow[dl, "j"] \\ & X & \end{tikzcd}$$

with homotopies

$$F : f \circ h_S \sim h_A \circ f'$$
$$G : g \circ h_S \sim h_B \circ g'$$
$$H : i \circ f \sim j \circ g,$$

and suppose that the bottom square is a pushout square, and the top squares are pullback squares. Then the type of tuples $((X', h_X), (i', I, p), (j', J, q), (H', C))$ consisting of

(i) A type $X' : \mathcal{U}$ together with a morphism

$$h_X : X' \to X,$$

(ii) A map $i' : A' \to X'$, a homotopy $I : i \circ h_A \sim h_X \circ i'$, and a term p witnessing that the square

$$
\begin{tikzcd}
A' \arrow[r, "i'"] \arrow[d, "h_A"'] & X' \arrow[d, "h_X"] \\
A \arrow[r, "i"'] & X
\end{tikzcd}
$$

is a pullback square.

(iii) A map $j' : B' \to X'$, a homotopy $J : j \circ h_B \sim h_X \circ j'$, and a term q witnessing that the square

$$
\begin{tikzcd}
B' \arrow[r, "j'"] \arrow[d, "h_B"'] & X' \arrow[d, "h_X"] \\
B \arrow[r, "j"'] & X
\end{tikzcd}
$$

is a pullback square,

(iv) A homotopy $H' : i' \circ f' \sim j' \circ g'$, and a homotopy

$$C : (i \cdot F) \cdot ((I \cdot f') \cdot (h_X \cdot H')) \sim (H \cdot h_S) \cdot ((j \cdot G) \cdot (J \cdot g'))$$

witnessing that the cube

$$
\begin{tikzcd}
& S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl] \arrow[dr] & B' \arrow[d] \arrow[dl] \\
A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\
& X, &
\end{tikzcd}
$$

commutes,

is contractible.

The following theorem should be compared to the flattening lemma, Lemma 24.3.2.

Theorem 24.5.6. *Consider a commuting cube*

$$
\begin{tikzcd}
& S' \arrow[dl, "f'"'] \arrow[d, "h_S"] \arrow[dr, "g'"] & \\
A' \arrow[d, "h_A"'] \arrow[dr, "i'"'] & S \arrow[dl, "f"'] \arrow[dr, "g"] & B' \arrow[d, "h_B"] \arrow[dl, "j'"] \\
A \arrow[dr, "i"'] & X' \arrow[d, "h_X"] & B \arrow[dl, "j"] \\
& X. &
\end{tikzcd}
$$

If each of the vertical squares is a pullback, and the bottom square is a pushout, then the top square is a pushout.

Proof. By Theorem 21.5.3 we have families of equivalences

$$\begin{aligned}
F &: \prod_{(x:S)} \mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_A}(f(x)) \\
G &: \prod_{(x:S)} \mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_B}(g(x)) \\
I &: \prod_{(a:A)} \mathsf{fib}_{h_A}(a) \simeq \mathsf{fib}_{h_X}(i(a)) \\
J &: \prod_{(b:B)} \mathsf{fib}_{h_B}(b) \simeq \mathsf{fib}_{h_X}(j(b)).
\end{aligned}$$

Moreover, since the cube commutes we obtain a family of homotopies

$$K : \prod_{(x:S)} I(f(x)) \circ F(x) \sim J(g(x)) \circ G(x).$$

We define the descent data (P_A, P_B, P_S) consisting of $P_A : A \to \mathcal{U}$, $P_B : B \to \mathcal{U}$, and $P_S : \prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x))$ by

$$\begin{aligned}
P_A(a) &:\equiv \mathsf{fib}_{h_A}(a) \\
P_B(b) &:\equiv \mathsf{fib}_{h_B}(b) \\
P_S(x) &:\equiv G(x) \circ F(x)^{-1}.
\end{aligned}$$

We have

$$\begin{aligned}
P &:\equiv \mathsf{fib}_{h_X} \\
e_A &:\equiv I \\
e_B &:\equiv J \\
e_S &:\equiv K.
\end{aligned}$$

Now consider the diagram

$$\begin{tikzcd}
\sum_{(s:S)} \mathsf{fib}_{h_S}(s) \arrow[r] \arrow[d] & \sum_{(s:S)} \mathsf{fib}_{h_A}(f(s)) \arrow[r] \arrow[d] & \sum_{(b:B)} \mathsf{fib}_{h_B}(b) \arrow[d] \\
\sum_{(a:A)} \mathsf{fib}_{h_A}(a) \arrow[r] & \sum_{(a:A)} \mathsf{fib}_{h_A}(a) \arrow[r] & \sum_{(x:X)} \mathsf{fib}_{h_X}(x)
\end{tikzcd}$$

Since the top and bottom map in the left square are equivalences, we obtain from Exercise 22.1 that the left square is a pushout square. Moreover, the right square is a pushout by Lemma 24.3.2. Therefore it follows by Theorem 22.3.4 that the outer rectangle is a pushout square.

Now consider the commuting cube

$$\begin{tikzcd}
& & \sum_{(s:S)} \mathsf{fib}_{h_S}(s) \arrow[dll] \arrow[d] \arrow[drr] & & \\
\sum_{(a:A)} \mathsf{fib}_{h_A}(a) \arrow[d] \arrow[drr] & & S' \arrow[dll] \arrow[drr] & & \sum_{(b:B)} \mathsf{fib}_{h_B}(b) \arrow[dll] \arrow[d] \\
A' \arrow[drr] & & \sum_{(x:X)} \mathsf{fib}_{h_X}(x) \arrow[d] & & B' \arrow[dll] \\
& & X'. & &
\end{tikzcd}$$

We have seen that the top square is a pushout. The vertical maps are all equivalences, so the vertical squares are all pushout squares. Thus it follows from one more application of Theorem 22.3.4 that the bottom square is a pushout. □

Theorem 24.5.7. *Consider a commuting cube of types*

$$\begin{tikzcd}
& S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\
& X, &
\end{tikzcd}$$

and suppose the vertical squares are pullback squares. Then the commuting square

$$\begin{tikzcd}
A' \sqcup^{S'} B' \arrow[r] \arrow[d] & X' \arrow[d] \\
A \sqcup^{S} B \arrow[r] & X
\end{tikzcd}$$

is a pullback square.

Proof. It suffices to show that the pullback

$$(A \sqcup^{S} B) \times_X X'$$

has the universal property of the pushout. This follows by the descent theorem, since the vertical squares in the cube

$$\begin{tikzcd}
& S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & (A \sqcup^{S} B) \times_X X' \arrow[d] & B \arrow[dl] \\
& A \sqcup^{S} B &
\end{tikzcd}$$

are pullback squares by Theorem 21.5.8. □

24.6 Applications of the descent theorem

Theorem 24.6.1. *Consider a commuting cube*

$$\begin{tikzcd}
 & S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl] \arrow[dr] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & C' \arrow[d] & B \arrow[dl] \\
 & C &
\end{tikzcd}$$

in which the bottom square is a pushout square. If the vertical sides are pullback squares, then for each $c : C$ the square of fibers

$$\begin{tikzcd}
\mathsf{fib}_{i\circ f\circ h_S}(c) \arrow[r] \arrow[d] & \mathsf{fib}_{j\circ g\circ h_S}(c) \arrow[r] & \mathsf{fib}_{j\circ h_B}(c) \arrow[d] \\
\mathsf{fib}_{i\circ h_A}(c) \arrow[rr] & & \mathsf{fib}_{h_C}(c)
\end{tikzcd}$$

is a pushout square.

Exercises

24.1 Use the characterization of the circle as a pushout given in Example 22.3.3 to show that the square

$$\begin{tikzcd}
\mathbf{S}^1 + \mathbf{S}^1 \arrow[r, "{[\mathsf{id},\mathsf{id}]}"] \arrow[d, "{[\mathsf{id},\mathsf{id}]}"'] & \mathbf{S}^1 \arrow[d, "{\lambda t.\,(t,\mathsf{base})}"] \\
\mathbf{S}^1 \arrow[r, "{\lambda t.\,(t,\mathsf{base})}"'] & \mathbf{S}^1 \times \mathbf{S}^1
\end{tikzcd}$$

is a pushout square.

24.2 Let $f : A \to B$ be a map. The **codiagonal** ∇_f of f is the map obtained from the universal property of the pushout, as indicated in the diagram

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "f"'] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\mathsf{id}_B"] & \\
A \arrow[r, "\mathsf{inl}"] \arrow[urphantom, phantom, ""] \arrow[drr, bend right, "\mathsf{id}_B"'] & B \sqcup^A B \arrow[dr, dotted, "\nabla_f"] & \\
 & & B
\end{tikzcd}$$

Show that $\mathsf{fib}_{\nabla_f}(b) \simeq \Sigma(\mathsf{fib}_f(b))$ for any $b : B$.

24.3 Consider two maps $f : A \to X$ and $g : B \to X$. The **fiberwise join** $f * g$ is defined by the universal property of the pushout as the unique map rendering the diagram

$$\begin{tikzcd} A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "g"] \\ A \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "f"'] & A *_X B \arrow[dr, dotted, "f*g"] \arrow[ul, phantom, "\ulcorner"] \\ & & X \end{tikzcd}$$

commutative, where $A *_X B$ is defined as a pushout, as indicated. Construct an equivalence

$$\mathsf{fib}_{f*g}(x) \simeq \mathsf{fib}_f(x) * \mathsf{fib}_g(x)$$

for any $x : X$.

24.4 Consider two maps $f : A \to B$ and $g : C \to D$. The **pushout-product**

$$f \Box g : (A \times D) \sqcup^{A \times C} (B \times C) \to B \times D$$

of f and g is defined by the universal property of the pushout as the unique map rendering the diagram

$$\begin{tikzcd} A \times C \arrow[r, "f \times \mathsf{id}_C"] \arrow[d, "\mathsf{id}_A \times g"'] & B \times C \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\mathsf{id}_B \times g"] \\ A \times D \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "f \times \mathsf{id}_D"'] & (A \times D) \sqcup^{A \times C} (B \times C) \arrow[dr, dotted, "f \Box g"] \\ & & B \times D \end{tikzcd}$$

commutative. Construct an equivalence

$$\mathsf{fib}_{f \Box g}(b, d) \simeq \mathsf{fib}_f(b) * \mathsf{fib}_g(d)$$

for all $b : B$ and $d : D$.

24.5 Let A and B be pointed types with base points $a_0 : A$ and $b_0 : B$. The **wedge inclusion** is defined as follows by the universal property of the wedge:

$$\begin{tikzcd} \mathbf{1} \arrow[r] \arrow[d] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\lambda b.\,(a_0, b)"] \\ A \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "\lambda a.\,(a, b_0)"'] & A \vee B \arrow[dr, dotted, "\mathsf{wedge\text{-}in}_{A,B}"] \\ & & A \times B \end{tikzcd}$$

Show that the fiber of the wedge inclusion $A \vee B \to A \times B$ is equivalent to $\Omega(B) * \Omega(A)$.

24.6 Let $f : X \vee X \to X$ be the map defined by the universal property of the wedge as indicated in the diagram

$$\begin{tikzcd} \mathbf{1} \arrow[r, "x_0"] \arrow[d, "x_0"'] & X \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\mathsf{id}_X"] \\ X \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "\mathsf{id}_X"'] & X \vee X \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dotted, "f"] \\ & & X. \end{tikzcd}$$

(a) Show that $\mathsf{fib}_f(x_0) \simeq \Sigma\Omega(X)$.
(b) Show that $\mathsf{cof}_f \simeq \Sigma X$.

24.7 Consider a pushout square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X, \end{tikzcd}$$

and suppose that f is an embedding. Show that j is an embedding, and that the square is also a pullback square.

24.8 Consider a pushout square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X. \end{tikzcd}$$

(a) Show that if f is surjective, then so is j.
(b) Show that the two small squares in the diagram

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "q_f"'] & B \arrow[d, "q_j"] \\ \mathrm{im}(f) \arrow[r, dotted] \arrow[d, "i_f"'] & \mathrm{im}(j) \arrow[d, "i_j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

are both pushout squares, and that the bottom square is also a pullback square.

25 The identity types of pushouts

25.1 Characterizing families of maps over pushouts

Definition 25.1.1. Consider a span $\mathcal{S}$

$$A \xleftarrow{f} S \xrightarrow{g} B,$$

and consider $P, Q : \mathsf{Fam\text{-}pushout}(\mathcal{S})$. A morphism of descent data from P to Q over $\mathcal{S}$ is defined to be a triple (h_A, h_B, h_S) consisting of

$$h_A : \prod_{(x:A)} P_A(x) \to Q_A(x)$$

$$h_B : \prod_{(y:B)} P_B(y) \to Q_B(y)$$

equipped with a homotopy h_S witnessing that the square

$$\begin{tikzcd} P_A(f(s)) \arrow[r, "h_A(f(s))"] \arrow[d, "P_S(s)"'] & Q_A(f(s)) \arrow[d, "Q_S(s)"] \\ P_B(g(s)) \arrow[r, "h_B(g(s))"'] & Q_B(g(s)) \end{tikzcd}$$

commutes for every $s : S$. We write $\mathsf{hom}_{\mathcal{S}}(P, Q)$ for the type of morphisms of descent data over $\mathcal{S}$.

An equivalence of descent data from P to Q is a morphism h such that h_A and h_B are families of equivalences.

Remark 25.1.2. The identity type $h = h'$ of $\mathsf{hom}_{\mathcal{S}}(P, Q)$ is characterized as the type of triples (H_A, H_B, H_S) consisting of

$$H_A : \prod_{(a:A)} h_A(a) \sim h'_A(a)$$
$$H_B : \prod_{(b:B)} h_B(b) \sim h'_B(b)$$

and a homotopy $K_S(s)$ witnessing that the square

$$\begin{tikzcd} h_B(g(s)) \circ P_S(s) \arrow[r] \arrow[d] & Q_S(s) \circ h_A(f(s)) \arrow[d] \\ h'_B(g(s)) \circ P_S(s) \arrow[r] & Q_S(s) \circ h'_A(g(s)) \end{tikzcd}$$

of homotopies commutes for every $s : S$.

Definition 25.1.3. Consider a commuting square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

with $H : i \circ f \sim j \circ f$, and let P and Q be type families over X. We define a map

$$\Big(\prod_{(x:X)} P(x) \to Q(x)\Big) \to \mathsf{hom}_{\mathcal{S}}(\mathsf{desc\text{-}fam}(P), \mathsf{desc\text{-}fam}(Q)).$$

Construction. Let $h : \prod_{(x:X)} P(x) \to Q(x)$. Then we define

$$h_A : \prod_{(a:A)} P(i(a)) \to Q(i(a))$$
$$h_B : \prod_{(b:B)} P(j(b)) \to Q(j(b))$$

by $h_A(a,p) :\equiv h(i(a),p)$ and $h_B(b,q) :\equiv h(j(b),q)$. Then it remains to define for every $s : S$ a homotopy $h_S(s)$ witnessing that the square

$$\begin{tikzcd}
P(i(f(s))) \arrow[r, "h_A(f(s))"] \arrow[d, "\mathsf{tr}_P(H(s))"'] & Q(i(f(s))) \arrow[d, "\mathsf{tr}_Q(H(s))"] \\
P(j(g(s))) \arrow[r, "h_B(g(s))"'] & Q(j(g(s)))
\end{tikzcd}$$

commutes. Note that every family of maps $h : \prod_{(x:X)} P(x) \to Q(x)$ is natural in the sense that for any path $p : x = x'$ in X, there is a homotopy $\psi(p,h)$ witnessing that the square

$$\begin{tikzcd}
P(x) \arrow[r, "h(x)"] \arrow[d, "\mathsf{tr}_P(p)"'] & Q(x) \arrow[d, "\mathsf{tr}_Q(p)"] \\
P(x') \arrow[r, "h(x')"'] & Q(x')
\end{tikzcd}$$

commutes. Therefore we define $h_S(s) :\equiv \psi(H(s),h)$. □

Theorem 25.1.4. *The map defined in Definition 25.1.3 is an equivalence.*

Proof. We will first construct a commuting triangle

$$\begin{tikzcd}
& \prod_{(x:X)} P(x) \to Q(x) \arrow[dl] \arrow[dr] & \\
\mathsf{dep\text{-}cocone}_{(i,j,H)}(x \mapsto P(x) \to Q(x)) \arrow[rr, dotted] & & \mathsf{hom}_{\mathcal{S}}(\mathsf{desc\text{-}fam}(P), \mathsf{desc\text{-}fam}(Q))
\end{tikzcd}$$

Recall from Theorem 24.1.4 that X satisfies the dependent universal property, so the map on the left is an equivalence. Therefore we will prove the claim by showing that the bottom map is an equivalence.

In order to construct the bottom map, we first note that for any two maps $\alpha : P(x) \to Q(x)$ and $\alpha' : P(x') \to Q(x')$ and any path $p : x = x'$, there is an equivalence

$$\varphi(p,f,f') : \Big(\mathsf{tr}_{x \mapsto P(x) \to Q(x)}(p,f) = f'\Big) \simeq \Big(\prod_{(y:B(x))} f'(\mathsf{tr}_B(p,y)) = \mathsf{tr}_C(p,f(y))\Big).$$

The equivalence φ is defined by path induction on p, where we take

$$\varphi(\mathsf{refl},f,f') :\equiv \mathsf{htpy\text{-}eq} \circ \mathsf{inv}.$$

Now we define the bottom map in the asserted triangle to be the map

$$(h_A, h_B, h_S) \mapsto (h_A, h_B, \lambda s.\, \varphi(H(s), h_A(f(s)), h_B(g(s)), h_S(s))).$$

Note that this map is an equivalence, since it is the induced map on total spaces of an equivalence.

It remains to show that the triangle commutes. By Remark 25.1.2 it suffices to construct families of homotopies

$$K_A : \prod_{(a:A)} h_{i(a)} \sim h_{i(a)}$$

$$K_B : \prod_{(b:B)} h_{j(b)} \sim h_{j(b)}$$

and for each $s : S$ a homotopy $K_S(s)$ witnessing that the square

$$\begin{tikzcd}
h_{j(g(s))} \circ \mathsf{tr}_P(H(s)) \arrow[rr, "\psi(H(s),h)"] \arrow[d] & & \mathsf{tr}_Q(H(s)) \circ h_{i(f(s))} \arrow[d] \\
h_{j(g(s))} \circ \mathsf{tr}_P(H(s)) \arrow[rr, "\varphi(H(s),h_{i(f(s))},h_{j(g(s))},\mathsf{apd}_h(H(s)))"'] & & \mathsf{tr}_Q(H(s)) \circ h_{i(f(s))}
\end{tikzcd}$$

commutes. Of course, we take $K_A(a) :\equiv \mathsf{htpy\text{-}refl}$ and $K_B(b) :\equiv \mathsf{htpy\text{-}refl}$, so it suffices to show that

$$\psi(H(s),h) \sim \varphi(H(s),h_{i(f(s))},h_{j(g(s))},\mathsf{apd}_h(H(s))).$$

Now we would like to proceed by homotopy induction on $H : i \circ f \sim j \circ g$. However, we can only do so after we generalize the problem sufficiently to a situation where H has free endpoints. It is indeed possible by homotopy induction to construct for every $f, g : S \to X$ equipped with a homotopy $H : f \sim g$, every family of maps $h : \prod_{(x:X)} P(x) \to Q(x)$ and every $s : S$, a homotopy

$$\psi(H(s),h) \sim \varphi(H(s),h_{f(s)},h_{g(s)},\mathsf{apd}_h(H(s))).$$ □

25.2 Characterizing the identity types of pushouts

Definition 25.2.1. Consider a span $\mathcal{S}$ equipped with $a : A$, and consider $P : \mathsf{Fam\text{-}pushout}(\mathcal{S})$ equipped with $p : P_A(a)$. We say that P is **universal** if for every $Q : \mathsf{Fam\text{-}pushout}(\mathcal{S})$ the evaluation map

$$\mathsf{hom}_{\mathcal{S}}(P,Q) \to Q_A(a)$$

given by $h \mapsto h_A(a,p)$ is an equivalence.

Lemma 25.2.2. *Consider a pushout square*

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\
A \arrow[r, "i"'] & X
\end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, and let $a : A$. Furthermore, let P be the descent data for the type family $x \mapsto i(a) = x$ over X. Then P is universal.

Proof. Since $\mathsf{desc\text{-}fam}$ is an equivalence, it suffices to show that for every type family Q over X, the map

$$\mathsf{hom}_{\mathcal{S}}(\mathsf{desc\text{-}fam}(\mathsf{Id}(i(a))),\mathsf{desc\text{-}fam}(Q)) \to Q(i(a))$$

given by $h \mapsto h_A(a, \mathsf{refl}_{i(a)})$ is an equivalence. Note that we have a commuting triangle

$$\begin{tikzcd}
\prod_{(x:X)}(i(a) = x) \to Q(x) \arrow[r] \arrow[dr, "\mathsf{ev\text{-}refl}"'] & \mathsf{hom}_{\mathcal{S}}(\mathsf{desc\text{-}fam}(\mathsf{Id}(i(a))),\mathsf{desc\text{-}fam}(Q)) \arrow[d, "h\mapsto h_A(\mathsf{refl}_{i(a)})"] \\
& Q(i(a))
\end{tikzcd}$$

The map $\mathsf{ev\text{-}refl}$ is an equivalence by Theorem 11.3.3, and the top map is an equivalence by Theorem 25.1.4. Therefore it follows that the remaining map is an equivalence. □

Theorem 25.2.3. *Consider a pushout square*

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

with $H : i \circ f \sim j \circ g$, and let $a : A$. Furthermore consider a pair (P, p_0) consisting of P : $\mathsf{Fam\text{-}pushout}(\mathcal{S})$ and $p : P_A(a)$. If P is universal, then we have two families of equivalences

$$e_A : \prod_{(x:A)} P_A(x) \simeq (i(a) = i(x))$$
$$e_B : \prod_{(y:B)} P_B(y) \simeq (i(a) = j(b))$$

equipped with a homotopy e_S witnessing that the square

$$\begin{tikzcd} P_A(f(s)) \arrow[r, "e(s)"] \arrow[d, "e_A(f(s))"'] & P_B(g(s)) \arrow[d, "e_B(g(s))"] \\ (i(a) = i(f(s))) \arrow[r, "\lambda p.\, p \cdot H(s)"'] & (i(a) = g(s)) \end{tikzcd}$$

commutes for each $s : S$, and an identification $e_A(a, r) = \mathsf{refl}_{i(a)}$

Theorem 25.2.4. *Let X be a pointed type with base point $x_0 : X$. Then the loop space of ΣX is the initial type Y equipped with a base point $y_0 : Y$, and a pointed map*

$$X \to_* (Y \simeq Y).$$

Proof. The type of pairs (Y, μ) consisting of a pointed type Y and a pointed map $\mu : X \to_* (Y \simeq Y)$ is equivalent to the type of triples (Y, Z, μ) consisting of a pointed type Y, a type Z, and a map $\mu : X \to (Y \simeq Z)$. □

Corollary 25.2.5. *The loop space of $\mathbf{S}^2$ is the initial type X equipped with a point $x_0 : X$ and a homotopy $H : \mathsf{id} \sim \mathsf{id}$.*

Exercises

25.1 Consider the suspension

$$\begin{tikzcd} P \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{S}"] \\ \mathbf{1} \arrow[r, "\mathsf{N}"'] & \Sigma P \end{tikzcd}$$

of a proposition P. Show that $(\mathsf{N} = \mathsf{S}) \simeq P$.

25.2 Show that if X has decidable equality, then ΣX is a 1-type.

25.3 Consider a pushout square

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & \mathbf{1} \arrow[d, "j"] \\ B \arrow[r, "i"'] & X \end{tikzcd}$$

where $f : A \to B$ is an embedding.

(a) Show that there are equivalences

$$(i(b) = i(y)) \simeq (b = y) * \mathsf{fib}_f(b)$$
$$(i(b) = j(\star)) \simeq \mathsf{fib}_f(b)$$

for any $b, y : B$.

(b) Use Exercise 25.4.b to show that if B is a k-type, then so is X, for any $k \geq 0$.

25.4 Consider the join

$$\begin{tikzcd} P \times X \arrow[r, "\mathsf{pr}_2"] \arrow[d, "\mathsf{pr}_1"'] & X \arrow[d, "\mathsf{inr}"] \\ P \arrow[r, "\mathsf{inl}"'] & P * X \end{tikzcd}$$

of a proposition P and an arbitrary type X.

(a) Show that for any $x, y : X$ there is an equivalence $e : (\mathsf{inr}(x) = \mathsf{inr}(y)) \simeq P * (x = y)$ for which the triangle

$$\begin{tikzcd} & (x = y) \arrow[dl, "\mathsf{ap}_{\mathsf{inr}}"'] \arrow[dr, "\mathsf{inr}"] & \\ (\mathsf{inr}(x) = \mathsf{inr}(y)) \arrow[rr, "e"'] & & P * (x = y) \end{tikzcd}$$

commutes.

(b) Show that if X is a k-type, then so is $P * X$.

26 The real projective spaces

26.1 The type of 2-element sets

Theorem 26.1.1. *The type*

$$\sum_{(X:\mathcal{U}_2)} X$$

is contractible.

Corollary 26.1.2. *For any 2-element type X, the map*

$$(\mathbf{2} = X) \to X$$

given by $p \mapsto \mathsf{tr}_{\mathcal{T}}(p, 1_{\mathbf{2}})$ is an equivalence.

26.2 Classifying real line bundles

26.3 The finite dimensional real projective spaces

27 Sequential colimits

Note: This chapter currently contains only the statements of the definitions and theorems, but no proofs. I hope to make a complete version available soon.

27.1 The universal property of sequential colimits

Type sequences are diagrams of the following form.

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots .$$

Their formal specification is as follows.

Definition 27.1.1. An **(increasing) type sequence** $\mathcal{A}$ consists of

$$A : \mathbb{N} \to \mathcal{U}$$
$$f : \prod_{(n:\mathbb{N})} A_n \to A_{n+1}.$$

In this section we will introduce the sequential colimit of a type sequence. The sequential colimit includes each of the types A_n, but we also identify each $x : A_n$ with its value $f_n(x) : A_{n+1}$. Imagine that the type sequence $A_0 \to A_1 \to A_2 \to \cdots$ defines a big telescope, with A_0 sliding into A_1, which slides into A_2, and so forth.

As usual, the sequential colimit is characterized by its universal property.

Definition 27.1.2. (i) A **(sequential) cocone** on a type sequence $\mathcal{A}$ with vertex B consists of

$$h : \prod_{(n:\mathbb{N})} A_n \to B$$
$$H : \prod_{(n:\mathbb{N})} h_n \sim h_{n+1} \circ f_n.$$

We write $\mathsf{cocone}(B)$ for the type of cocones with vertex B.

(ii) Given a cocone (h, H) with vertex B on a type sequence $\mathcal{A}$ we define the map

$$\mathsf{cocone\text{-}map}(h, H) : (B \to C) \to \mathsf{cocone}(C)$$

given by $f \mapsto (\lambda n.\, f \circ h_n, \lambda n.\, \lambda x.\, \mathsf{ap}_f(H_n(x)))$.

(iii) We say that a cocone (h, H) with vertex B is **colimiting** if $\mathsf{cocone\text{-}map}(h, H)$ is an equivalence for any type C.

Theorem 27.1.3. *Consider a cocone (h, H) with vertex B for a type sequence $\mathcal{A}$. The following are equivalent:*

(i) The cocone (h, H) is colimiting.

(ii) The cocone (h, H) is inductive in the sense that for every type family $P : B \to \mathcal{U}$, the map

$$\Big(\prod_{(b:B)} P(b)\Big) \to \sum_{(h:\prod_{(n:\mathbb{N})} \prod_{(x:A_n)} P(h_n(x)))} \prod_{(n:\mathbb{N})} \prod_{(x:A_n)} \mathsf{tr}_P(H_n(x), h_n(x)) = h_{n+1}(f_n(x))$$

given by

$$s \mapsto (\lambda n.\, s \circ h_n, \lambda n.\, \lambda x.\, \mathsf{apd}_s(H_n(x)))$$

has a section.

(iii) The map in (ii) is an equivalence.

27.2 The construction of sequential colimits

We construct sequential colimits using pushouts.

Definition 27.2.1. Let $\mathcal{A} \equiv (A, f)$ be a type sequence. We define the type A_∞ as a pushout

$$\begin{tikzcd} \tilde{A} + \tilde{A} \arrow[r, "{[\mathsf{id}, \sigma_{\mathcal{A}}]}"] \arrow[d, "{[\mathsf{id}, \mathsf{id}]}"'] & \tilde{A} \arrow[d, "\mathsf{inr}"] \\ \tilde{A} \arrow[r, "\mathsf{inl}"'] & A_\infty. \end{tikzcd}$$

Definition 27.2.2. The type A_∞ comes equipped with a cocone structure consisting of

$$\begin{aligned} \mathsf{seq\text{-}in} &: \prod_{(n:\mathbb{N})} A_n \to A_\infty \\ \mathsf{seq\text{-}glue} &: \prod_{(n:\mathbb{N})} \prod_{(x:A_n)} \mathsf{in}_n(x) = \mathsf{in}_{n+1}(f_n(x)). \end{aligned}$$

Construction. We define

$$\begin{aligned} \mathsf{seq\text{-}in}(n, x) &:\equiv \mathsf{inr}(n, x) \\ \mathsf{seq\text{-}glue}(n, x) &:\equiv \mathsf{glue}(\mathsf{inl}(n, x))^{-1} \cdot \mathsf{glue}(\mathsf{inr}(n, x)). \end{aligned}$$

□

Theorem 27.2.3. *Consider a type sequence $\mathcal{A}$, and write $\tilde{A} :\equiv \sum_{(n:\mathbb{N})} A_n$. Moreover, consider the map*

$$\sigma_{\mathcal{A}} : \tilde{A} \to \tilde{A}$$

defined by $\sigma_{\mathcal{A}}(n, a) :\equiv (n + 1, f_n(a))$. Furthermore, consider a cocone (h, H) with vertex B. The following are equivalent:

(i) The cocone (h, H) with vertex B is colimiting.

(ii) The defining square

$$\begin{tikzcd} \tilde{A} + \tilde{A} \arrow[r, "{[\mathsf{id}, \sigma_{\mathcal{A}}]}"] \arrow[d, "{[\mathsf{id}, \mathsf{id}]}"'] & \tilde{A} \arrow[d, "\lambda(n,x).\, h_n(x)"] \\ \tilde{A} \arrow[r, "\lambda(n,x).\, h_n(x)"'] & A_\infty, \end{tikzcd}$$

of A_∞ is a pushout square.

27.3 Descent for sequential colimits

Definition 27.3.1. The type of **descent data** on a type sequence $\mathcal{A} \equiv (A, f)$ is defined to be

$$\mathsf{Desc}(\mathcal{A}) :\equiv \sum_{(B:\prod_{(n:\mathbb{N})} A_n \to \mathcal{U})} \prod_{(n:\mathbb{N})} \prod_{(x:A_n)} B_n(x) \simeq B_{n+1}(f_n(x)).$$

Definition 27.3.2. We define a map

$$\mathsf{desc\text{-}fam} : (A_\infty \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{A})$$

by $B \mapsto (\lambda n.\, \lambda x.\, B(\mathsf{seq\text{-}in}(n, x)), \lambda n.\, \lambda x.\, \mathsf{tr}_B(\mathsf{seq\text{-}glue}(n, x)))$.

Theorem 27.3.3. *The map*

$$\mathsf{desc\text{-}fam} : (A_\infty \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{A})$$

is an equivalence.

Definition 27.3.4. A **cartesian transformation** of type sequences from $\mathcal{A}$ to $\mathcal{B}$ is a pair (h, H) consisting of

$$\begin{aligned} h &: \textstyle\prod_{(n:\mathbb{N})} A_n \to B_n \\ H &: \textstyle\prod_{(n:\mathbb{N})} g_n \circ h_n \sim h_{n+1} \circ f_n, \end{aligned}$$

such that each of the squares in the diagram

$$\begin{tikzcd} A_0 \arrow[r, "f_0"] \arrow[d, "h_0"'] & A_1 \arrow[r, "f_1"] \arrow[d, "h_1"'] & A_2 \arrow[r, "f_2"] \arrow[d, "h_2"'] & \cdots \\ B_0 \arrow[r, "g_0"'] & B_1 \arrow[r, "g_1"'] & B_2 \arrow[r, "g_2"'] & \cdots \end{tikzcd}$$

is a pullback square. We define

$$\begin{aligned} \mathsf{cart}(\mathcal{A}, \mathcal{B}) :\equiv {} & \textstyle\sum_{(h:\prod_{(n:\mathbb{N})} A_n \to B_n)} \\ & \textstyle\sum_{(H:\prod_{(n:\mathbb{N})} g_n \circ h_n \sim h_{n+1} \circ f_n)} \prod_{(n:\mathbb{N})} \mathsf{is\text{-}pullback}(h_n, f_n, H_n), \end{aligned}$$

and we write

$$\mathsf{Cart}(\mathcal{B}) :\equiv \textstyle\sum_{(\mathcal{A}:\mathsf{Seq})} \mathsf{cart}(\mathcal{A}, \mathcal{B}).$$

Definition 27.3.5. We define a map

$$\mathsf{cart\text{-}map}(\mathcal{B}) : \Big(\textstyle\sum_{(X':\mathcal{U})} X' \to X\Big) \to \mathsf{Cart}(\mathcal{B}).$$

which associates to any morphism $h : X' \to X$ a cartesian transformation of type sequences into $\mathcal{B}$.

Theorem 27.3.6. *The operation* $\mathsf{cart\text{-}map}(\mathcal{B})$ *is an equivalence.*

27.4 The flattening lemma for sequential colimits

The flattening lemma for sequential colimits essentially states that sequential colimits commute with Σ.

Lemma 27.4.1. *Consider*

$$\begin{aligned} B &: \textstyle\prod_{(n:\mathbb{N})} A_n \to \mathcal{U} \\ g &: \textstyle\prod_{(n:\mathbb{N})} \prod_{(x:A_n)} B_n(x) \simeq B_{n+1}(f_n(x)). \end{aligned}$$

and suppose $P : A_\infty \to \mathcal{U}$ *is the unique family equipped with*

$$e : \textstyle\prod_{(n:\mathbb{N})} B_n(x) \simeq P(\mathsf{seq\text{-}in}(n, x))$$

and homotopies $H_n(x)$ witnessing that the square

$$\begin{tikzcd}
B_n(x) \arrow[r, "g_n(x)"] \arrow[d, "e_n(x)"'] & B_{n+1}(f_n(x)) \arrow[d, "e_{n+1}(f_n(x))"] \\
P(\mathsf{seq\text{-}in}(n,x)) \arrow[r, "\mathsf{tr}_P(\mathsf{seq\text{-}glue}(n,x))"'] & P(\mathsf{seq\text{-}in}(n+1,f_n(x)))
\end{tikzcd}$$

commutes. Then $\sum_{(t:A_\infty)} P(t)$ satisfies the universal property of the sequential colimit of the type sequence

$$\textstyle\sum_{(x:A_0)} B_0(x) \xrightarrow{\mathsf{tot}_{f_0}(g_0)} \sum_{(x:A_1)} B_1(x) \xrightarrow{\mathsf{tot}_{f_1}(g_1)} \sum_{(x:A_2)} B_2(x) \xrightarrow{\mathsf{tot}_{f_2}(g_2)} \cdots.$$

In the following theorem we rephrase the flattening lemma in using cartesian transformations of type sequences.

Theorem 27.4.2. *Consider a commuting diagram of the form*

$$\begin{tikzcd}
A_0 \arrow[r] \arrow[dd] \arrow[drr] & A_1 \arrow[rr] \arrow[dd] \arrow[dr] & & A_2 \arrow[r] \arrow[dd] \arrow[dl] & \cdots \arrow[dll] \\
& & X \arrow[dd] & & \\
B_0 \arrow[r] \arrow[drr] & B_1 \arrow[rr] \arrow[dr] & & B_2 \arrow[r] \arrow[dl] & \cdots \arrow[dll] \\
& & Y & &
\end{tikzcd}$$

If each of the vertical squares is a pullback square, and Y is the sequential colimit of the type sequence B_n, then X is the sequential colimit of the type sequence A_n.

Corollary 27.4.3. *Consider a commuting diagram of the form*

$$\begin{tikzcd}
A_0 \arrow[r] \arrow[dd] \arrow[drr] & A_1 \arrow[rr] \arrow[dd] \arrow[dr] & & A_2 \arrow[r] \arrow[dd] \arrow[dl] & \cdots \arrow[dll] \\
& & X \arrow[dd] & & \\
B_0 \arrow[r] \arrow[drr] & B_1 \arrow[rr] \arrow[dr] & & B_2 \arrow[r] \arrow[dl] & \cdots \arrow[dll] \\
& & Y & &
\end{tikzcd}$$

If each of the vertical squares is a pullback square, then the square

$$\begin{tikzcd}
A_\infty \arrow[r] \arrow[d] & X \arrow[d] \\
B_\infty \arrow[r] & Y
\end{tikzcd}$$

is a pullback square.

27.5 Constructing the propositional truncation

The propositional truncation can be used to construct the image of a map, so we construct that first. We construct the propositional truncation of A via a construction called the **join construction**, as the colimit of the sequence of join-powers of A

$$A \longrightarrow A * A \longrightarrow A * (A * A) \longrightarrow \cdots$$

The join-powers of A are defined recursively on n, by taking[3]

$$\begin{aligned} A^{*0} &:\equiv \emptyset \\ A^{*1} &:\equiv A \\ A^{*(n+2)} &:\equiv A * A^{*(n+1)}. \end{aligned}$$

We will write $A^{*\infty}$ for the colimit of the sequence

$$A \xrightarrow{\mathsf{inr}} A * A \xrightarrow{\mathsf{inr}} A * (A * A) \xrightarrow{\mathsf{inr}} \cdots .$$

The sequential colimit $A^{*\infty}$ comes equipped with maps $\mathsf{in\text{-}seq}_n : A^{*(n+1)} \to A^{*\infty}$, and we will write

$$\eta :\equiv \mathsf{in\text{-}seq}_0 : A \to A^{*\infty}.$$

Our goal is to show $A^{*\infty}$ is a proposition, and that $\eta : A \to A^{*\infty}$ satisfies the universal property of the propositional truncation of A. Before showing that $A^{*\infty}$ is indeed a proposition, let us show in two steps that for any proposition P, the map

$$(A^{*\infty} \to P) \to (A \to P)$$

is indeed an equivalence.

Lemma 27.5.1. *Suppose $f : A \to P$, where A is any type and P is a proposition. Then the precomposition function*

$$- \circ \mathsf{inr} : (A * B \to P) \to (B \to P)$$

is an equivalence, for any type B.

Proof. Since the precomposition function

$$- \circ \mathsf{inr} : (A * B \to P) \to (B \to P)$$

is a map between propositions, it suffices to construct a map

$$(B \to P) \to (A * B \to P).$$

Let $g : B \to P$. Then the square

$$\begin{array}{ccc} A \times B & \xrightarrow{\mathsf{pr}_2} & B \\ {\scriptstyle \mathsf{pr}_1}\downarrow & & \downarrow{\scriptstyle g} \\ A & \xrightarrow[f]{} & P \end{array}$$

[3] In this definition, the case $A^{*1} :\equiv A$ is slightly redundant because we have an equivalence

$$A * \emptyset \simeq A.$$

Nevertheless, it is nice to have that $A^{*1} \equiv A$.

commutes since P is a proposition. Therefore we obtain a map $A * B \to P$ by the universal property of the join. □

Proposition 27.5.2. *Let A be a type, and let P be a proposition. Then the function*

$$- \circ \eta : (A^{*\infty} \to P) \to (A \to P)$$

is an equivalence.

Proof. Since the map

$$- \circ \eta : (A^{*\infty} \to P) \to (A \to P)$$

is a map between propositions, it suffices to construct a map in the converse direction.

Let $f : A \to P$. First, we show by recursion that there are maps

$$f_n : A^{*(n+1)} \to P.$$

The map f_0 is of course just defined to be f. Given $f_n : A^{*(n+1)}$ we obtain $f_{n+1} : A * A^{*(n+1)} \to P$ by Lemma 27.5.1. Because P is assumed to be a proposition it is immediate that the maps f_n form a cocone with vertex P on the sequence

$$A \xrightarrow{\mathsf{inr}} A * A \xrightarrow{\mathsf{inr}} A * (A * A) \xrightarrow{\mathsf{inr}} \cdots.$$

From this cocone we obtain the desired map $(A^{*\infty} \to P)$. □

Proposition 27.5.3. *The type $A^{*\infty}$ is a proposition for any type A.*

Proof. By Theorem 10.1.3 it suffices to show that

$$A^{*\infty} \to \mathsf{is\text{-}contr}(A^{*\infty}).$$

Since the type $\mathsf{is\text{-}contr}(A^{*\infty})$ is already known to be a proposition by Exercise 11.2, it follows from Proposition 27.5.2 that it suffices to show that

$$A \to \mathsf{is\text{-}contr}(A^{*\infty}).$$

Let $x : A$. To see that $A^{*\infty}$ is contractible it suffices by Exercise 27.3 to show that $\mathsf{inr} : A^{*n} \to A^{*(n+1)}$ is homotopic to the constant function $\mathsf{const}_{\mathsf{inl}(x)}$. However, we get a homotopy $\mathsf{const}_{\mathsf{inl}(x)} \sim \mathsf{inr}$ immediately from the path constructor glue. □

All the definitions are now in place to define the propositional truncation of a type.

Definition 27.5.4. For any type A we define the type

$$\|A\|_{-1} :\equiv A^{*\infty},$$

and we define $\eta : A \to \|A\|_{-1}$ to be the constructor $\mathsf{in\text{-}seq}_0$ of the sequential colimit $A^{*\infty}$. Often we simply write $\|A\|$ for $\|A\|_{-1}$.

The type $\|A\|_{-1}$ is a proposition by Proposition 27.5.3, and

$$\eta : A \to \|A\|_{-1}$$

satisfies the universal property of propositional truncation by Proposition 27.5.2.

Proposition 27.5.5. *The propositional truncation operation is functorial in the sense that for any map $f : A \to B$ there is a unique map $\|f\| : \|A\| \to \|B\|$ such that the square*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\eta"'] & B \arrow[d, "\eta"] \\ \|A\| \arrow[r, "\|f\|"'] & \|B\| \end{tikzcd}$$

commutes. Moreover, there are homotopies

$$\|\mathsf{id}_A\| \sim \mathsf{id}_{\|A\|}$$
$$\|g \circ f\| \sim \|g\| \circ \|f\|.$$

Proof. The functorial action of propositional truncation is immediate by the universal property of propositional truncation. To see that the functorial action preserves the identity, note that the type of maps $\|A\| \to \|A\|$ for which the square

$$\begin{tikzcd} A \arrow[r, "\mathsf{id}"] \arrow[d, "\eta"'] & A \arrow[d, "\eta"] \\ \|A\| \arrow[r, dotted] & \|A\| \end{tikzcd}$$

commutes is contractible. Since this square commutes for both $\|\mathsf{id}\|$ and for id, it must be that they are homotopic. The proof that the functorial action of propositional truncation preserves composition is similar. □

27.6 Proving type theoretical replacement

Our goal is now to show that the image of a map $f : A \to B$ from an essentially small type A into a locally small type B is again essentially small. This property is called the type theoretic replacement property. In order to prove this property, we have to find another construction of the image of a map. In order to make this construction, we define a join operation on maps.

Definition 27.6.1. Consider two maps $f : A \to X$ and $g : B \to X$ with a common codomain X.

(i) We define the type $A *_X B$ as the pushout

$$\begin{tikzcd} A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & B \arrow[d, "\mathsf{inr}"] \\ A \arrow[r, "\mathsf{inl}"'] & A *_X B. \end{tikzcd}$$

(ii) We define the **join** $f * g : A *_X B \to X$ to be the unique map for which the diagram

$$\begin{tikzcd} A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "g"] & \\ A \arrow[r, "\mathsf{inl}"'] \arrow[drr, bend right, "f"'] & A *_X B \arrow[dr, dotted, "f*g"] & \\ & & X \end{tikzcd}$$

The reason to call the map $f * g$ the join of f and g is that the fiber of $f * g$ at any $x : X$ is equivalent to the join of the fibers of f and g at x.

Lemma 27.6.2. *Consider two maps $f : A \to X$ and $g : B \to X$. Then there is an equivalence*

$$\mathsf{fib}_{f*g}(x) \simeq \mathsf{fib}_f(x) * \mathsf{fib}_g(x)$$

for any $x : X$.

Proof. Consider the commuting cube

$$\begin{tikzcd}
 & & \mathsf{fib}_f(x) \times \mathsf{fib}_g(x) \arrow[dll] \arrow[d] \arrow[drr] & & \\
\mathsf{fib}_f(x) \arrow[d] \arrow[drr] & & A \times_X B \arrow[dll, crossing over] \arrow[drr, crossing over] & & \mathsf{fib}_g(x) \arrow[dll] \arrow[d] \\
A \arrow[drr] & & \mathbf{1} \arrow[d] & & B \arrow[dll] \\
 & & X & &
\end{tikzcd}$$

In this cube, the bottom square is a canonical pullback square. The two squares in the front are pullbacks by Lemma 21.4.1, and the top square is a pullback square by Lemma 21.3.1. Therefore it follows by Remark 23.1.5 that all the faces of this cube are pullback squares, and hence by Theorem 24.5.7 we obtain that the square

$$\begin{tikzcd}
\mathsf{fib}_f(x) * \mathsf{fib}_g(x) \arrow[r] \arrow[d, dotted] & \mathbf{1} \arrow[d] \\
A *_X B \arrow[r, "f*g"'] & X
\end{tikzcd}$$

is a pullback square. Now the claim follows by the uniqueness of pullbacks, which was shown in Corollary 21.1.8. $\square$

Lemma 27.6.3. *Consider a map $f : A \to X$, an embedding $m : U \to X$, and $h : \hom_X(f, m)$. Then the map*

$$\hom_X(f * g, m) \to \hom_X(g, m)$$

is an equivalence for any $g : B \to X$.

Proof. Note that both types are propositions, so any equivalence can be used to prove the claim. Thus, we simply calculate

$$\begin{aligned}
\hom_X(f * g, m) &\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_{f*g}(x) \to \mathsf{fib}_m(x) \\
&\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_f(x) * \mathsf{fib}_g(x) \to \mathsf{fib}_m(x) \\
&\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_g(x) \to \mathsf{fib}_m(x) \\
&\simeq \hom_X(g, m).
\end{aligned}$$

The first equivalence holds by Exercise 11.15; the second equivalence holds by Exercise 24.3, also using Theorem 11.4.1 and Exercise 11.3 where we established that that pre- and postcomposing by an equivalence is an equivalence; the third equivalence holds by Lemma 27.5.1 and Exercise 11.3; the last equivalence again holds by Exercise 11.15. □

For the construction of the image of $f : A \to X$ we observe that if we are given an embedding $m : U \to X$ and a map $(i, I) : \hom_X(f, m)$, then (i, I) extends uniquely along $\mathsf{inr} : A \to A *_X A$ to a map $\hom_X(f * f, m)$. This extension again extends uniquely along $\mathsf{inr} : A *_X A \to A *_X (A *_X A)$ to a map $\hom_X(f * (f * f), m)$ and so on, resulting in a diagram of the form

$$\begin{tikzcd} A \arrow[r, "\mathsf{inr}"] \arrow[dr] & A *_X A \arrow[r, "\mathsf{inr}"] \arrow[d, dotted] & A *_X (A *_X A) \arrow[r, "\mathsf{inr}"] \arrow[dl, dotted] & \cdots \arrow[dll, dotted, bend left] \\ & U & & \end{tikzcd}$$

Definition 27.6.4. Suppose $f : A \to X$ is a map. Then we define the **fiberwise join powers**

$$f^{*n} : A_X^{*n} X.$$

Construction. Note that the operation $(B, g) \mapsto (A *_X B, f * g)$ defines an endomorphism on the type

$$\textstyle\sum_{(B:\mathcal{U})} B \to X.$$

We also have $(\emptyset, \mathsf{ind}_\emptyset)$ and (A, f) of this type. For $n \geq 1$ we define

$$A_X^{*(n+1)} :\equiv A *_X A_X^{*n}$$
$$f^{*(n+1)} :\equiv f * f^{*n}.$$

□

Definition 27.6.5. We define $A_X^{*\infty}$ to be the sequential colimit of the type sequence

$$\begin{tikzcd} A_X^{*0} \arrow[r] & A_X^{*1} \arrow[r, "\mathsf{inr}"] & A_X^{*2} \arrow[r, "\mathsf{inr}"] & \cdots. \end{tikzcd}$$

Since we have a cocone

$$\begin{tikzcd} A_X^{*0} \arrow[r] \arrow[dr, "f^{*0}"'] & A_X^{*1} \arrow[r, "\mathsf{inr}"] \arrow[d, "f^{*1}"'] & A_X^{*2} \arrow[r, "\mathsf{inr}"] \arrow[dl, "f^{*2}"'] & \cdots \arrow[dll, bend left] \\ & X & & \end{tikzcd}$$

we also obtain a map $f^{*\infty} : A_X^{*\infty} \to X$ by the universal property of $A_X^{*\infty}$.

Lemma 27.6.6. *Let $f : A \to X$ be a map, and let $m : U \to X$ be an embedding. Then the function*

$$- \circ \mathsf{in\text{-}seq}_0 : \hom_X(f^{*\infty}, m) \to \hom_X(f, m)$$

is an equivalence.

Theorem 27.6.7. *For any map $f : A \to X$, the map $f^{*\infty} : A_X^{*\infty} \to X$ is an embedding that satisfies the universal property of the image inclusion of f.*

Lemma 27.6.8. *Consider a commuting square*

$$\begin{tikzcd} A \arrow[r] \arrow[d] & B \arrow[d] \\ C \arrow[r] & D. \end{tikzcd}$$

(i) *If the square is cartesian, B and C are essentially small, and D is locally small, then A is essentially small.*

(ii) *If the square is cocartesian, and A, B, and C are essentially small, then D is essentially small.*

Corollary 27.6.9. *Suppose $f : A \to X$ and $g : B \to X$ are maps from essentially small types A and B, respectively, to a locally small type X. Then $A \times_X B$ is again essentially small.*

Lemma 27.6.10. *Consider a type sequence*

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots$$

where each A_n is essentially small. Then its sequential colimit is again essentially small.

Theorem 27.6.11. *For any map $f : A \to B$ from an essentially small type A into a locally small type B, the image of f is again essentially small.*

Corollary 27.6.12. *Consider a $\mathcal{U}$-small type A, and an equivalence relation R over A valued in the $\mathcal{U}$-small propositions. Then the set quotient A/R is essentially small.*

Exercises

27.1 Show that the sequential colimit of a type sequence

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots$$

is equivalent to the sequential colimit of its shifted type sequence

$$A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \cdots .$$

27.2 Let $P_0 \longrightarrow P_1 \longrightarrow P_2 \longrightarrow \cdots$ be a sequence of propositions. Show that

$$\mathsf{colim}_n(P_n) \simeq \exists_{(n:\mathbb{N})} P_n.$$

27.3 Consider a type sequence

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots$$

and suppose that $f_n \sim \mathsf{const}_{a_{n+1}}$ for some $a_n : \prod_{(n:\mathbb{N})} A_n$. Show that the sequential colimit is contractible.

27.4 Define the ∞-sphere $\mathbf{S}^\infty$ as the sequential colimit of

$$\mathbf{S}^0 \xrightarrow{f_0} \mathbf{S}^1 \xrightarrow{f_1} \mathbf{S}^2 \xrightarrow{f_2} \cdots$$

where $f_0 : \mathbf{S}^0 \to \mathbf{S}^1$ is defined by $f_0(0_\mathbf{2}) \equiv \mathsf{inl}(\star)$ and $f_0(1_\mathbf{2}) \equiv \mathsf{inr}(\star)$, and $f_{n+1} : \mathbf{S}^{n+1} \to \mathbf{S}^{n+2}$ is defined as $\Sigma(f_n)$. Use Exercise 27.3 to show that $\mathbf{S}^\infty$ is contractible.

27.5 Consider a type sequence

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots$$

in which $f_n : A_n \to A_{n+1}$ is weakly constant in the sense that

$$\prod_{(x,y:A_n)} f_n(x) = f_n(y)$$

Show that A_∞ is a mere proposition.

27.6 Show that $\mathbb{N}$ is the sequential colimit of

$$\mathsf{Fin}(0) \xrightarrow{\mathsf{inl}} \mathsf{Fin}(1) \xrightarrow{\mathsf{inl}} \mathsf{Fin}(2) \xrightarrow{\mathsf{inl}} \cdots.$$

Chapter V

Synthetic homotopy theory

28 Homotopy groups of types

28.1 The suspension-loop space adjunction

We get an even better version of the universal property of ΣX if we know in advance that the type X is a pointed type: on pointed types, the suspension functor is left adjoint to the loop space functor. This property manifests itself in the setting of pointed types, so we first give some definitions regarding pointed types.

Definition 28.1.1. (i) A pointed type consists of a type X equipped with a base point $x : X$. We will write $\mathcal{U}_*$ for the type $\sum_{(X:\mathcal{U})} X$ of all pointed types.

(ii) Let $(X, *_X)$ be a pointed type. A **pointed family** over $(X, *_X)$ consists of a type family $P : X \to \mathcal{U}$ equipped with a base point $*_P : P(*_X)$.

(iii) Let $(P, *_P)$ be a pointed family over $(X, *_X)$. A **pointed section** of $(P, *_P)$ consists of a dependent function $f : \prod_{(x:X)} P(x)$ and an identification $p : f(*_X) = *_P$. We define the **pointed Π-type** to be the type of pointed sections:

$$\prod^*_{(x:X)} P(x) :\equiv \sum_{(f:\prod_{(x:X)} P(x))} f(*_X) = *_P$$

In the case of two pointed types X and Y, we may also view Y as a pointed family over X. In this case we write $X \to_* Y$ for the type of pointed functions.

(iv) Given any two pointed sections f and g of a pointed family P over X, we define the type of pointed homotopies

$$f \sim_* g :\equiv \Pi^*_{(x:X)} f(x) = g(x),$$

where the family $x \mapsto f(x) = g(x)$ is equipped with the base point $p \cdot q^{-1}$.

Remark 28.1.2. Since pointed homotopies are defined as certain pointed sections, we can use the same definition of pointed homotopies again to consider pointed homotopies between pointed homotopies, and so on.

Example 28.1.3. For any type X, the suspension ΣX is a pointed type where the base point is taken to be the north pole $\mathbb{N}$.

Definition 28.1.4. Let X be a pointed type with base point x. We define the **loop space** $\Omega(X, x)$ of X at x to be the pointed type $x = x$ with base point refl_x.

Definition 28.1.5. The loop space operation Ω is *functorial* in the sense that

(i) For every pointed map $f : X \to_* Y$ there is a pointed map

$$\Omega(f) : \Omega(X) \to_* \Omega(Y),$$

defined by $\Omega(f)(\omega) :\equiv p_f \cdot \mathsf{ap}_f(\omega) \cdot p_f^{-1}$, which is base point preserving by $\mathsf{right\text{-}inv}(p_f)$.

(ii) For every pointed type X there is a pointed homotopy

$$\Omega(\mathsf{id}_X^*) \sim_* \mathsf{id}_{\Omega(X)}^*.$$

(iii) For any two pointed maps $f : X \to_* Y$ and $g : Y \to_* X$, there is a pointed homotopy witnessing that the triangle

$$\begin{tikzcd} & \Omega(Y) \arrow[dr, "\Omega(g)"] & \\ \Omega(X) \arrow[ur, "\Omega(f)"] \arrow[rr, "\Omega(g \circ_* f)"'] & & \Omega(Z) \end{tikzcd}$$

of pointed types commutes.

In order to introduce the suspension-loop space adjunction, we also need to construct the functorial action of suspension.

Definition 28.1.6. (i) Given a pointed map $f : X \to_* Y$, we define a map

$$\Sigma(f) : \Sigma X \to_* \Sigma Y$$

Definition 28.1.7. We define a pointed map

$$\varepsilon_X : X \to_* \Omega(\Sigma X)$$

for any pointed type X. This map is called the **counit** of the suspension-loop space adjunction. Moreover, ε is natural in X in the sense that for any pointed map $f : X \to_* Y$ we have a commuting square

$$\begin{tikzcd} X \arrow[r, "\varepsilon_X"] \arrow[d, "f"'] & \Omega(\Sigma X) \arrow[d, "\Omega(\Sigma f)"] \\ Y \arrow[r, "\varepsilon_X"'] & \Omega(\Sigma Y) \end{tikzcd}$$

Construction. The underlying map of ε_X takes $x : X$ to the concatenation

$$\mathsf{N} \xlongequal{\mathsf{merid}(x)} \mathsf{S} \xlongequal{\mathsf{merid}(*_X)^{-1}} \mathsf{N}.$$

This map preserves the base point, since $\mathsf{merid}(*_X) \cdot \mathsf{merid}(*_X)^{-1} = \mathsf{refl}_\mathsf{N}$. $\square$

Definition 28.1.8. (i) For any pointed type X, we define the **pointed identity function** $\mathsf{id}_X^* :\equiv (\mathsf{id}_X, \mathsf{refl}_*)$.

(ii) For any two pointed maps $f : X \to_* Y$ and $g : Y \to_* Z$, we define the **pointed composite**

$$g \circ_* f :\equiv (g \circ f, \mathsf{ap}_g(p_f) \cdot p_g).$$

Definition 28.1.9. Given two pointed types X and Y, a pointed map from X to Y is a pair (f, p) consisting of a map $f : X \to Y$ and a path $p : f(x_0) = y_0$ witnessing that f preserves the base point. We write

$$X \to_* Y$$

for the type of **pointed maps** from X to Y. The type $X \to_* Y$ is itself a pointed type, with base point $(\mathsf{const}_{y_0}, \mathsf{refl}_{y_0})$.

Now suppose that we have a pointed map $f : \Sigma X \to_* Y$ with $p : f(x_0) = y_0$. Then the composite

$$X \xrightarrow{\varepsilon_X} \Omega(\Sigma X) \xrightarrow{\Omega(f)} \Omega(Y)$$

yields a pointed map $\tilde{f} : X \to \Omega(Y)$. Therefore we obtain a map

$$\tau_{X,Y} : (\Sigma X \to_* Y) \to (X \to_* \Omega(Y)).$$

It is not hard to see that also $\tau_{X,Y}$ is pointed. We leave this to the reader. The following theorem is also called the adjointness of the suspension and loop space functors. This is an extremely important relation that pops up in many calculations of homotopy groups.

Theorem 28.1.10. *Let X and Y be pointed types. Then the pointed map*

$$\tau_{X,Y} : (\Sigma X \to_* Y) \to_* (X \to_* \Omega(Y))$$

is an equivalence. Moreover, τ is pointedly natural in X and Y.

28.2 Homotopy groups

Definition 28.2.1. For $n \geq 1$, the **n-th homotopy group** of a type X at a base point $x : X$ consists of the type

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

equipped with the group operations inherited from the path operations on $\Omega^n(X, x)$. Often we will simply write $\pi_n(X)$ when it is clear from the context what the base point of X is.

For $n \equiv 0$ we define $\pi_0(X, x) :\equiv \|X\|_0$.

Example 28.2.2. In **??** we established that $\Omega(\mathbf{S}^1) \simeq \mathbb{Z}$. It follows that

$$\pi_1(\mathbf{S}^1) = \mathbb{Z} \qquad \text{and} \qquad \pi_n(\mathbf{S}^1) = 0 \qquad \text{for } n \geq 2.$$

Furthermore, we have seen in **??** that $\|\mathbf{S}^1\|_0$ is contractible. Therefore we also have $\pi_0(\mathbf{S}^1) = 0$.

28.3 The Eckmann-Hilton argument

Given a diagram of identifications

$$\begin{tikzcd} x \arrow[r, equal, bend left=60, "p"] \arrow[r, equal, "p'"] \arrow[r, equal, bend right=60, "p''"'] \arrow[r, phantom, shift left=3, "r \Downarrow"] \arrow[r, phantom, shift right=3, "r' \Downarrow"] & y \end{tikzcd}$$

in a type A, where $r : p = p'$ and $r' : p' = p''$, we obtain by concatenation an identification $r \cdot r' : p = p''$. This operation on identifications of identifications is sometimes called the **vertical concatenation**, because there is also a *horizontal* concatenation operation.

Definition 28.3.1. Consider identifications of identifications $r : p = p'$ and $s : q = q'$, where $p, p' : x = y$, and $q, q' : y = z$ are identifications in a type A, as indicated in the diagram

$$\begin{tikzcd} x \arrow[r, bend left, equal, "p"] \arrow[r, bend right, equal, "p'"'] \arrow[r, phantom, "r \Downarrow"] & y \arrow[r, bend left, equal, "q"] \arrow[r, bend right, equal, "q'"'] \arrow[r, phantom, "s \Downarrow"] & z. \end{tikzcd}$$

We define the **horizontal concatenation** $r \cdot_h s : p \cdot q = p' \cdot q'$ of r and s.

Proof. First we induct on r, so it suffices to define $\mathsf{refl}_p \cdot_h s : p \cdot q = p \cdot q'$. Next, we induct on p, so it suffices to define $\mathsf{refl}_{\mathsf{refl}_y} \cdot_h s : \mathsf{refl}_y \cdot q = \mathsf{refl}_y \cdot q'$. Since $\mathsf{refl}_y \cdot q \equiv q$ and $\mathsf{refl}_y \cdot q' \equiv q'$, we take $\mathsf{refl}_{\mathsf{refl}_y} \cdot_h s :\equiv s$. □

Lemma 28.3.2. *Horizontal concatenation satisfies the left and right unit laws.*

In the following lemma we establish the **interchange law** for horizontal and vertical concatenation.

Lemma 28.3.3. *Consider a diagram of the form*

$$\begin{tikzcd}[column sep=huge] x \arrow[r, bend left=60, equal, "p"] \arrow[r, equal] \arrow[r, bend right=60, equal, "p''"'] \arrow[r, phantom, shift left=3ex, "r \Downarrow"] \arrow[r, phantom, shift right=3ex, "r' \Downarrow"] & y \arrow[r, bend left=60, equal, "q"] \arrow[r, equal] \arrow[r, bend right=60, equal, "q''"'] \arrow[r, phantom, shift left=3ex, "s \Downarrow"] \arrow[r, phantom, shift right=3ex, "s' \Downarrow"] & z. \end{tikzcd}$$

Then there is an identification

$$(r \cdot r') \cdot_h (s \cdot s') = (r \cdot_h s) \cdot (r' \cdot_h s').$$

Proof. We use path induction on both r and r', followed by path induction on p. Then it suffices to show that

$$(\mathsf{refl}_{\mathsf{refl}_y} \cdot \mathsf{refl}_{\mathsf{refl}_y}) \cdot_h (s \cdot s') = (\mathsf{refl}_{\mathsf{refl}_y} \cdot_h s) \cdot (\mathsf{refl}_{\mathsf{refl}_y} \cdot_h s').$$

Using the computation rules, we see that this reduces to

$$s \cdot s' = s \cdot s',$$

which we have by reflexivity. □

Theorem 28.3.4. *For $n \geq 2$, the n-th homotopy group is abelian.*

Proof. Our goal is to show that

$$\prod_{(r,s:\pi_2(X))} r \cdot s = s \cdot r.$$

Since we are constructing an identification in a set, we can use the universal property of 0-truncation on both r and s. Therefore it suffices to show that

$$\prod_{(r,s:\mathsf{refl}_{x_0} = \mathsf{refl}_{x_0})} |r|_0 \cdot |s|_0 = |s|_0 \cdot |r|_0.$$

Now we use that $|r|_0 \cdot |s|_0 \equiv |r \cdot s|_0$ and $|s|_0 \cdot |r|_0 \equiv |s \cdot r|_0$, to see that it suffices to show that $r \cdot s = s \cdot r$, for every $r, s : \mathsf{refl}_x = \mathsf{refl}_x$. Using the unit laws and the interchange law, this is a simple computation:

$$\begin{aligned} r \cdot s &= (r \cdot_h \mathsf{refl}_x) \cdot (\mathsf{refl}_x \cdot_h s) \\ &= (r \cdot \mathsf{refl}_x) \cdot_h (\mathsf{refl}_x \cdot s) \\ &= (\mathsf{refl}_x \cdot r) \cdot_h (s \cdot \mathsf{refl}_x) \\ &= (\mathsf{refl}_x \cdot_h s) \cdot (r \cdot_h \mathsf{refl}_x) \\ &= s \cdot r. \end{aligned}$$

□

Exercises

28.1 Show that the type of pointed families over a pointed type (X, x) is equivalent to the type

$$\textstyle\sum_{(Y:\mathcal{U}_*)} Y \to_* X.$$

28.2 Given two pointed types A and X, we say that A is a (pointed) retract of X if we have $i : A \to_* X$, a retraction $r : X \to_* A$, and a pointed homotopy $H : r \circ_* i \sim_* \mathsf{id}^*$.

(a) Show that if A is a pointed retract of X, then $\Omega(A)$ is a pointed retract of $\Omega(X)$.
(b) Show that if A is a pointed retract of X and $\pi_n(X)$ is a trivial group, then $\pi_n(A)$ is a trivial group.

28.3 Construct by path induction a family of maps

$$\prod_{(A,B:\mathcal{U})} \prod_{(a:A)} \prod_{(b:B)} ((A, a) = (B, b)) \to \textstyle\sum_{(e:A \simeq B)} e(a) = b,$$

and show that this map is an equivalence. In other words, an *identification of pointed types* is a base point preserving equivalence.

28.4 Let (A, a) and (B, b) be two pointed types. Construct by path induction a family of maps

$$\prod_{(f,g:A \to B)} \prod_{(p:f(a)=b)} \prod_{(q:g(a)=b)} ((f, p) = (g, q)) \to \textstyle\sum_{(H:f \sim g)} p = H(a) \cdot q,$$

and show that this map is an equivalence. In other words, an *identification of pointed maps* is a base point preserving homotopy.

28.5 Show that if $A \leftarrow S \to B$ is a span of pointed types, then for any pointed type X the square

$$\begin{array}{ccc} (A \sqcup^S B \to_* X) & \longrightarrow & (B \to_* X) \\ \downarrow & & \downarrow \\ (A \to_* X) & \longrightarrow & (S \to_* X) \end{array}$$

is a pullback square.

28.6 Let $f : A \to_* B$ be a pointed map. Show that the following are equivalent:

(i) f is an equivalence.
(ii) For any pointed type X, the precomposition map

$$- \circ_* f : (B \to_* X) \to_* (A \to_* X)$$

is an equivalence.

28.7 In this exercise we prove the suspension-loopspace adjunction.

(a) Construct a pointed equivalence

$$\tau_{X,Y} : (\Sigma(X) \to_* Y) \simeq_* (X \to \Omega(Y))$$

for any two pointed spaces X and Y.

(b) Show that for any $f : X \to_* X'$ and $g : Y' \to_* Y$, there is a pointed homotopy witnessing that the square

$$\begin{tikzcd} (\Sigma(X') \to_* Y') \arrow[r, "\tau_{X',Y'}"] \arrow[d, "h \mapsto g \circ h \circ \Sigma(f)"'] & (X' \to_* \Omega(Y')) \arrow[d, "h \mapsto \Omega(g) \circ h \circ f"] \\ (\Sigma(X) \to_* Y) \arrow[r, "\tau_{X,Y}"'] & (X \to_* \Omega(Y)) \end{tikzcd}$$

28.8 Show that if

$$\begin{tikzcd} C \arrow[r] \arrow[d] & B \arrow[d] \\ A \arrow[r] & X \end{tikzcd}$$

is a pullback square of pointed types, then so is

$$\begin{tikzcd} \Omega(C) \arrow[r] \arrow[d] & \Omega(B) \arrow[d] \\ \Omega(A) \arrow[r] & \Omega(X). \end{tikzcd}$$

28.9 (a) Show that if X is k-truncated, then its n-th homotopy group $\pi_n(X)$ is trivial for each choice of base point, and each $n > k$.

(b) Show that if X is $(k + l)$-truncated, and for each $0 < i \leq l$ the $(k + i)$-th homotopy groups $\pi_{k+i}(X)$ are trivial for each choice of base point, then X is k-truncated.

It is consistent to assume that there are types for which all homotopy groups are trivial, but which aren't contractible nonetheless. Such types are called **∞-connected**.

28.10 Consider a cospan

$$A \xrightarrow{f} X \xleftarrow{g} B$$

of pointed types and pointed maps between them.

(a) Define the type of pointed cones $\mathsf{cone}_*(C)$, where the vertex C is a pointed type. Also characterize its identity type.

(b) Define for any pointed cone (p, q, H) with vertex C the map

$$\mathsf{cone\text{-}map}_*(p, q, H) : (C' \to_* C) \to \mathsf{cone}_*(C').$$

Now we can say that the cone (p, q, H) satisfies the universal property of the pointed pullback of the cospan $A \to X \leftarrow A$ if this map is an equivalence for each pointed type C'.

(c) Now consider a commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X, \end{tikzcd}$$

where f and g are assumed to be pointed maps between pointed types (they come equipped with $\alpha : f(a_0) = x_0$ and $\beta : g(b_0) = x_0$, respectively). Show that if C is a pullback (in the usual unpointed sense), then C can be given the structure of a pointed pullback in a unique way, i.e., show that the type of

$$\begin{aligned}
&c_0 : C \\
&\gamma : p(c_0) = a_0 \\
&\delta : q(c_0) = b_0 \\
&\varepsilon : \mathsf{ap}_f(\gamma) \cdot \alpha = H(c_0) \cdot (\mathsf{ap}_g(\delta) \cdot \beta)
\end{aligned}$$

for which C satisfies the universal property of a pointed pullback, is contractible.

(d) Conclude that a commuting square of pointed types is a pointed pullback square if and only if the underlying square of unpointed types is an ordinary pullback square.

29 The Hopf fibration

Our goal in this section is to construct the **Hopf fibration**. The Hopf fibration is a fiber sequence

$$\mathbf{S}^1 \hookrightarrow \mathbf{S}^3 \twoheadrightarrow \mathbf{S}^2.$$

More generally, we show that for any type A equipped with a multiplicative operation $\mu : A \to (A \to A)$ for which $\mu(x, -)$ and $\mu(-, x)$ are equivalences, there is a fiber sequence

$$A \hookrightarrow A * A \twoheadrightarrow \Sigma A.$$

The construction of this fiber sequence is known as the **Hopf construction**. We then get the Hopf fibration from the Hopf construction by using the multiplication on $\mathbf{S}^1$ constructed in §18.3 after we show that $\mathbf{S}^1 * \mathbf{S}^1 \simeq \mathbf{S}^3$.

We then introduce the long exact sequence of homotopy groups. The long exact sequence is an important tool to compute homotopy groups which applies to any fiber sequence

$$F \hookrightarrow E \twoheadrightarrow B.$$

In the case of the Hopf fibration, we will use the long exact sequence to show that

$$\pi_k(\mathbf{S}^3) = \pi_k(\mathbf{S}^2)$$

for any $k \geq 3$.

Since the Hopf fibration is closely related to the multiplication operation of the complex numbers on the unit circle, the Hopf fibration is sometimes also called the *complex* Hopf fibration. Indeed, there is also a *real* Hopf fibration

$$\mathbf{S}^0 \hookrightarrow \mathbf{S}^1 \twoheadrightarrow \mathbf{S}^1.$$

This is just the double cover of the circle. There is even a *quaternionic* Hopf fibration

$$\mathbf{S}^3 \hookrightarrow \mathbf{S}^7 \twoheadrightarrow \mathbf{S}^4,$$

which uses the multiplication of the quaternionic numbers on the unit sphere. The main difficulty in defining the quaternionic Hopf fibration in homotopy type theory is to define the quaternionic multiplication

$$\mathsf{mul}_{\mathbf{S}^3} : \mathbf{S}^3 \to (\mathbf{S}^3 \to \mathbf{S}^3).$$

The construction of the octonionic Hopf fibration

$$\mathbf{S}^7 \hookrightarrow \mathbf{S}^{15} \twoheadrightarrow \mathbf{S}^8$$

in homotopy type theory is still an open problem. Another open problem is to formalize Adams' theorem [1] in homotopy type theory, that there are *no* further fiber sequences of the form

$$\mathbf{S}^k \hookrightarrow \mathbf{S}^l \twoheadrightarrow \mathbf{S}^m,$$

for $k, l, m \geq 0$.

29.1 Fiber sequences

Definition 29.1.1. A **short sequence** of maps into a pointed type B with base point b consists of maps

$$F \xrightarrow{i} E \xrightarrow{p} B$$

equipped with a homotopy $p \circ i \sim \mathsf{const}_b$. We say that a short sequence as above is an **unpointed fiber sequence** if the commuting square

$$\begin{tikzcd} F \arrow[r, "i"] \arrow[d, "\mathsf{const}_\star"'] & E \arrow[d, "p"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_b"'] & B \end{tikzcd}$$

is a pullback square.

Definition 29.1.2. A **short sequence** of pointed maps into a pointed type B with base point b consists of pointed maps

$$F \xrightarrow{i} E \xrightarrow{p} B$$

equipped with a pointed homotopy $p \circ i \sim_* \mathsf{const}_b$. We say that a short sequence as above is an **fiber sequence** if the commuting square

$$\begin{tikzcd} F \arrow[r, "i"] \arrow[d, "\mathsf{const}_\star"'] & E \arrow[d, "p"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_b"'] & B \end{tikzcd}$$

is a pullback square.

29.2 The Hopf construction

The Hopf construction is a general construction of a fiber sequence

$$A \hookrightarrow A * A \twoheadrightarrow \Sigma A,$$

that applies to any H-space A. Our definition of an H-space is chosen such that it provides only the necessary structure to apply the Hopf construction. We give an unpointed and a pointed variant, and moreover we give a coherent variant that is more closely related to the traditional definition of an H-space.

Definition 29.2.1.

(i) An **unpointed H-space** structure on a type A consists of a multiplicative operation

$$\mu : A \to (A \to A)$$

such that $\mu(x, -)$ and $\mu(-, x)$ are equivalences, for each $x : A$.

(ii) If A is a pointed type with base point $e : A$, then an **H-space** structure on A is an unpointed H-space structure on A equipped with an identification $\mu(e, e) = e$.

(iii) A **coherent H-space** structure on a pointed type A with base point $e : A$ consists of an unpointed H-space structure μ on A that satisfies the unit laws, i.e., μ comes equipped with identifications

$$\begin{aligned} \mathsf{left\text{-}unit}_\mu &: \mu(e, a) = a \\ \mathsf{right\text{-}unit}_\mu &: \mu(a, e) = a \\ \mathsf{coh\text{-}unit}_\mu &: \mathsf{left\text{-}unit}_\mu(e) = \mathsf{right\text{-}unit}_\mu(e). \end{aligned}$$

Example 29.2.2. The loop space $\Omega(A)$ of any pointed type is a coherent H-space, where the multiplication is given by path concatenation.

By an unpointed fiber sequence, we mean a sequence

$$F \xhookrightarrow{i} E \xrightarrow{p} B$$

where only the type B is assumed to be pointed (with base point b), and the square

$$\begin{tikzcd} F \arrow[r, "i"] \arrow[d, "\mathsf{const}_\star"'] & E \arrow[d, "p"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_b"'] & B \end{tikzcd}$$

is a pullback square.

Theorem 29.2.3 (The Hopf construction)**.** *Consider a type A equipped with an H-space structure μ. Then there is an unpointed fiber sequence*

$$A \hookrightarrow A * A \twoheadrightarrow \Sigma A.$$

If A and the H-space structure are pointed, then this unpointed fiber sequence is an fiber sequence.

Proof. Note that there is a unique map $h : A * A \to \Sigma A$ such that the cube

$$\begin{tikzcd} & & A \times A \arrow[dll, "\mathsf{pr}_1"'] \arrow[d, "\mu"'] \arrow[drr, "\mathsf{pr}_2"] & & \\ A \arrow[d] \arrow[drr] & & A \arrow[dll] \arrow[drr] & & A \arrow[d] \arrow[dll] \\ \mathbf{1} \arrow[drr] & & A * A \arrow[d, dotted, "h"'] & & \mathbf{1} \arrow[dll] \\ & & \Sigma A & & \end{tikzcd}$$

commutes. Thus we see that we obtain a fiber sequence $A \hookrightarrow A * A \twoheadrightarrow \Sigma A$ if we show that the front two squares are pullback squares. By the descent theorem, Theorem 24.4.1, it suffices to show that the two squares in the back

$$\begin{tikzcd} A \times A \arrow[r, "\mu"] \arrow[d, "\mathsf{pr}_1"'] & A \arrow[d] \\ A \arrow[r] & \mathbf{1} \end{tikzcd} \qquad \begin{tikzcd} A \times A \arrow[r, "\mu"] \arrow[d, "\mathsf{pr}_2"'] & A \arrow[d] \\ A \arrow[r] & \mathbf{1} \end{tikzcd}$$

are pullback squares. We claim that in both squares, the multiplicative operation μ induces equivalences on the fibers, and hence both squares are pullbacks by Theorem 21.5.3. To see this, note that the induced map on fibers fit in commuting squares

$$\begin{tikzcd} \mathsf{fib}_{\mathsf{pr}_1}(x) \arrow[r, dotted] \arrow[d, "\simeq"'] & \mathsf{fib}_{\mathsf{const}_\star}(\star) \arrow[d, "\simeq"] \\ A \arrow[r, "\mu(x,-)"'] & A \end{tikzcd} \qquad \begin{tikzcd} \mathsf{fib}_{\mathsf{pr}_2}(x) \arrow[r, dotted] \arrow[d, "\simeq"'] & \mathsf{fib}_{\mathsf{const}_\star}(\star) \arrow[d, "\simeq"] \\ A \arrow[r, "\mu(-,x)"'] & A. \end{tikzcd}$$

The claim now follows, since we have assumed that $\mu(x, -)$ and $\mu(-, x)$ are equivalences for each $x : X$. □

Remark 29.2.4. The Hopf map h constructed in Theorem 29.2.3 is the unique map $A * A \to \Sigma A$ equipped with identifications

$$p : \mathsf{N} = h(\mathsf{inl}(x))$$
$$p' : \mathsf{S} = h(\mathsf{inr}(x'))$$

and an identification q witnessing that the square

$$\begin{tikzcd} \mathsf{N} \arrow[r, equal, "p"] \arrow[d, equal, "\mathsf{merid}(\mu(x,x'))"'] & h(\mathsf{inl}(x)) \arrow[d, equal, "\mathsf{ap}_h(\mathsf{glue}(x,x'))"] \\ \mathsf{S} \arrow[r, equal, "p'"'] & h(\mathsf{inr}(x')) \end{tikzcd}$$

commutes.

Corollary 29.2.5. *There is a fiber sequence*

$$\mathbf{S}^1 \hookrightarrow \mathbf{S}^1 * \mathbf{S}^1 \twoheadrightarrow \mathbf{S}^2.$$

Proof. By Theorem 29.2.3 it suffices to construct an H-space structure on $\mathbf{S}^1$. This H-space structure $\mathbf{S}^1 \times \mathbf{S}^1 \to \mathbf{S}^1$ is determined by the complex multiplication operation constructed in Definition 18.3.1. □

Lemma 29.2.6. *The join operation is associative*

Proof.

$$\begin{tikzcd} A & A \times C \arrow[l] \arrow[r] & A \times C \\ A \times B \arrow[u] \arrow[d] & A \times B \times C \arrow[l] \arrow[r] \arrow[u] \arrow[d] & A \times C \arrow[u] \arrow[d] \\ B & B \times C \arrow[l] \arrow[r] & C \end{tikzcd}$$

□

Corollary 29.2.7. *There is an equivalence* $\mathbf{S}^1 * \mathbf{S}^1 \simeq \mathbf{S}^3$.

Theorem 29.2.8. *There is a fiber sequence* $\mathbf{S}^1 \hookrightarrow \mathbf{S}^3 \twoheadrightarrow \mathbf{S}^2$.

Lemma 29.2.9. *Suppose $f : G \to H$ is a group homomorphism, such that the sequence*

$$0 \longrightarrow G \xrightarrow{\ f\ } H \longrightarrow 0$$

is exact at G and H, where we write 0 for the trivial group consisting of just the unit element. Then f is a group isomorphism.

Corollary 29.2.10. *We have $\pi_2(\mathbf{S}^2) = \mathbb{Z}$, and for $k > 2$ we have $\pi_k(\mathbf{S}^2) = \pi_k(\mathbf{S}^3)$.*

29.3 The long exact sequence

Definition 29.3.1. A fiber sequence $F \hookrightarrow E \twoheadrightarrow B$ consists of:

(i) Pointed types F, E, and B, with base points x_0, y_0, and b_0 respectively,

(ii) Base point preserving maps $i : F \to_* E$ and $p : E \to_* B$, with $\alpha : i(x_0) = y_0$ and $\beta : p(y_0) = b_0$,

(iii) A pointed homotopy $H : \mathsf{const}_{b_0} \sim_* p \circ_* i$ witnessing that the square

$$\begin{tikzcd} F \arrow[r, "i"] \arrow[d] & E \arrow[d, "p"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_{b_0}"'] & B, \end{tikzcd}$$

commutes and is a pullback square.

Lemma 29.3.2. *Any fiber sequence $F \hookrightarrow E \twoheadrightarrow B$ induces a sequence of pointed maps*

$$\Omega(F) \xrightarrow{\Omega(i)} \Omega(E) \xrightarrow{\Omega(p)} \Omega(B) \xrightarrow{\ \partial\ } F \xrightarrow{\ i\ } E \xrightarrow{\ p\ } B,$$

in which every two consecutive maps form a fiber sequence.

Proof. By taking pullback squares repeatedly, we obtain the diagram

$$\begin{tikzcd} \Omega(F) \arrow[r] \arrow[d, "\Omega(i)"'] & \mathbf{1} \arrow[d, "\mathsf{const}_{\mathsf{refl}_{b_0}}"] & \\ \Omega(E) \arrow[r, "\Omega(p)"] \arrow[d] & \Omega(B) \arrow[r] \arrow[d, "\partial"'] & \mathbf{1} \arrow[d, "\mathsf{const}_{y_0}"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_{x_0}"'] & F \arrow[r, "i"] \arrow[d] & E \arrow[d, "p"] \\ & \mathbf{1} \arrow[r, "\mathsf{const}_{b_0}"'] & B. \end{tikzcd}$$

□

Definition 29.3.3. We say that a consecutive pair of pointed maps between pointed sets

$$A \xrightarrow{f} B \xrightarrow{g} C$$

is **exact** at B if we have

$$\left(\exists_{(a:A)} f(a) = b\right) \leftrightarrow (g(b) = c)$$

for any $b : B$.

Remark 29.3.4. If a pair of consecutive pointed maps between pointed sets

$$A \xrightarrow{f} B \xrightarrow{g} C$$

is exact at B, it directly that $\mathrm{im}(f) = \mathsf{fib}_g(c)$. Indeed, such a pair of pointed maps is exact at B if and only if there is an equivalence $e : \mathrm{im}(f) \simeq \mathsf{fib}_g(c)$ such that the triangle

$$\begin{tikzcd} \mathrm{im}(f) \arrow[rr, "e"] \arrow[dr] & & \mathsf{fib}_g(c) \arrow[dl] \\ & B & \end{tikzcd}$$

commutes. In other words, $\mathrm{im}(f)$ and $\mathsf{fib}_g(c)$ are equal *as subsets of* B.

Lemma 29.3.5. *Suppose $F \hookrightarrow E \twoheadrightarrow B$ is a fiber sequence. Then the sequence*

$$\|F\|_0 \xrightarrow{\|i\|_0} \|E\|_0 \xrightarrow{\|p\|_0} \|B\|_0$$

is exact at $\|E\|_0$.

Proof. To show that the image $\mathrm{im}\,\|i\|_0$ is the fiber $\mathsf{fib}_{\|p\|_0}(|b_0|_0)$, it suffices to construct a fiberwise equivalence

$$\prod_{(x:\|E\|_0)} \|\mathsf{fib}_{\|i\|_0}(x)\|_{-1} \simeq \|p\|_0(x) = |b_0|_0.$$

By the universal property of 0-truncation it suffices to show that

$$\prod_{(x:E)} \|\mathsf{fib}_{\|i\|_0}(|x|_0)\|_{-1} \simeq \|p\|_0(|x|_0) = |b_0|_0.$$

First we note that

$$\begin{aligned} \|p\|_0(|x|_0) = |b_0|_0 &\simeq |p(x)|_0 = |b_0|_0 \\ &\simeq \|p(x) = b_0\|_{-1}. \end{aligned}$$

Next, we note that

$$\begin{aligned} \mathsf{fib}_{\|i\|_0}(|x|_0) &\simeq \textstyle\sum_{(y:\|F\|_0)} \|i\|_0(y) = |x|_0 \\ &\simeq \|\textstyle\sum_{(y:F)} \|i\|_0(|y|_0) = |x|_0\|_0 \\ &\simeq \|\textstyle\sum_{(y:F)} |i(y)|_0 = |x|_0\|_0 \\ &\simeq \|\textstyle\sum_{(y:F)} \|i(y) = x\|_{-1}\|_0. \end{aligned}$$

Therefore it follows that

$$\begin{aligned}\|\mathsf{fib}_{\|i\|_0}(|x|_0)\|_{-1} &\simeq \|\textstyle\sum_{(y:F)}\|i(y) = x\|_{-1}\|_{-1}\\ &\simeq \|\textstyle\sum_{(y:F)} i(y) = x\|_{-1}\end{aligned}$$

Now it suffices to show that $\left(\sum_{(y:F)} i(y) = x\right) \simeq p(x) = b_0$. This follows by the pasting lemma of pullbacks

$$\begin{tikzcd}
(p(x) = b_0) \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\
F \arrow[r] \arrow[d] & E \arrow[d] \\
\mathbf{1} \arrow[r] & B
\end{tikzcd}$$

□

Theorem 29.3.6. *Any fiber sequence $F \hookrightarrow E \twoheadrightarrow B$ induces a long exact sequence on homotopy groups*

$$\begin{tikzcd}
 & & \cdots \arrow[dll, rounded corners] \\
\pi_n(F) \arrow[r, "\pi_n(i)"] & \pi_n(E) \arrow[r, "\pi_n(p)"] & \pi_n(B) \arrow[dll, dotted, rounded corners] \\
\pi_1(F) \arrow[r, "\pi_1(i)"] & \pi_1(E) \arrow[r, "\pi_1(p)"] & \pi_1(B) \arrow[dll, rounded corners] \\
\pi_0(F) \arrow[r, "\pi_0(i)"] & \pi_0(E) \arrow[r, "\pi_0(p)"] & \pi_0(B)
\end{tikzcd}$$

29.4 The universal complex line bundle

Definition 29.4.1. A **coherently associative unpointed H-space** structure on a type X consists of

29.5 The finite dimensional complex projective spaces

Remark 29.5.1. The universe of types that are merely equal to the circle does not classify complex line bundles.

Exercises

29.1 Consider an unpointed H-space X of which the multiplication is associative, and consider $x : X$. Construct a unit for the multiplication, and show that it satisfies the coherent unit laws.

29.2 (a) Show that the type of associative unpointed H-space structures on $\mathbf{2}$ is equivalent to $\mathbf{2}$.
(b) Show that the type of associative (pointed) H-space structures on $(\mathbf{2}, 1_{\mathbf{2}})$ is contractible.

29.3 Show that any fiber sequence

$$F \hookrightarrow E \twoheadrightarrow B$$

where the base points are $x_0 : B$, $y_0 : F$, and $z_0 : E$ induces a fiber sequence of connected components

$$\mathsf{BAut}(y_0) \hookrightarrow \mathsf{BAut}(z_0) \twoheadrightarrow \mathsf{BAut}(x_0).$$

29.4 Show that there is a fiber sequence

$$\mathbf{S}^3 \hookrightarrow \mathbf{S}^2 \twoheadrightarrow \|\mathbf{S}^2\|_2,$$

where the map $\mathbf{S}^2 \to \|\mathbf{S}^2\|_2$ is the unit of the 2-truncation.

29.5 Show that $\mathbb{C}\mathsf{P}^\infty$ is a coherent H-space. Note: the 2-sphere is not an H-space, and yet its 2-truncation is!

29.6 Construct for every group G of order $n+1$ a fiber sequence

$$G \longhookrightarrow \bigvee_{(i:\mathsf{Fin}(n^2))} \mathbf{S}^1 \longtwoheadrightarrow \bigvee_{(i:\mathsf{Fin}(n))} \mathbf{S}^1$$

29.7 Show that there is a fiber sequence

$$\mathbb{R}\mathsf{P}^\infty \hookrightarrow \mathbb{C}\mathsf{P}^\infty \twoheadrightarrow \mathbb{C}\mathsf{P}^\infty.$$

29.8 Show that the type of (small) fiber sequences is equivalent to the type of quadruples (B, P, b_0, x_0), consisting of

$$\begin{aligned} B &: \mathcal{U} \\ P &: B \to \mathcal{U} \\ b_0 &: B \\ x_0 &: P(b_0). \end{aligned}$$

30 Truncations

Truncation is a universal way of turning an arbitrary type into a k-truncated type. We have already seen the propositional truncation of a type X in §14, which is the proposition that X is merely inhabited, and the set truncation of X in §16.4, which is the set of connected components of X. The k-truncation is a generalization of the propositional truncation and the set truncation to an arbitrary truncation level k.

We construct the truncations by recursion on k. The base case $k \equiv -2$ is just the operation that sends a type X to the unit type $\mathbf{1}$, because up to equivalence there is only one contractible type. For the inductive step, we need to construct the $(k+1)$-truncation assuming that the k-truncation of an arbitrary type in a fixed universe $\mathcal{U}$ exists. Our construction of the $(k+1)$-truncation is a direct generalization of the construction of the set truncation as a set quotient, where we quotient out the equivalence relation

$$(x \sim y) :\equiv \|x = y\|_{-1}.$$

The idea is simple: if $\|X\|_{k+1}$ is to be the universal $(k+1)$-truncated type equipped with a map $|-|_{k+1} : X \to \|X\|_{k+1}$, then it has to be the case that

$$(|x|_{k+1} = |x'|_{k+1}) \simeq \|x = y\|_k.$$

We prove that this is indeed the case in Corollary 30.2.7.

The construction of the $(k+1)$-truncation as a quotient is different than the construction of the $(k+1)$-truncation that appears in [3] as a higher inductive type. This construction is based on the observation that a type X is k-truncated if and only if every map $\mathbf{S}^{k+1} \to X$ is constant. In other words, for every map $f : \mathbf{S}^{k+1} \to X$ into a k-type X, there is a point $x : X$ and a family of paths $p(t) : x = f(t)$. If we think of f as a 'wheel' in X, then x is the hub at the center of the wheel, and the paths $p(t)$ are the spokes. This leads to defining the k-truncation of a type X by the *hubs-and-spokes method*. In §30.3 we show that the k-truncation of a type is such a higher inductive type.

30.1 The universal property of the truncations

Definition 30.1.1. Let X be a type. A map $f : X \to Y$ into an k-type Y is said to satisfy the **universal property of the k-truncation of** X if the precomposition map

$$- \circ f : (Y \to Z) \to (X \to Z)$$

is an equivalence for every k-type Z.

Remark 30.1.2. A map $f : X \to Y$ into an k-type Y satisfies the universal property of k-truncation if of for every $g : X \to Z$ the type of extensions

$$\begin{tikzcd} X \arrow[d, "f"'] \arrow[dr, "g"] & \\ Y \arrow[r, dotted] & Z \end{tikzcd}$$

is contractible. Indeed, the type of such extensions is the type

$$\textstyle\sum_{(h:Y\to Z)} h \circ f \sim g,$$

which is equivalent to the fiber of the precomposition map $- \circ f$ at g.

In the following proposition we show that if a map $f : X \to Y$ into a k-type Y satisfies the universal property of the k-truncation of X, then f also satisfies the dependent elimination property.

Proposition 30.1.3. *Suppose the map $f : X \to Y$ into an k-type Y. The following are equivalent:*

(i) The map f satisfies the universal property of k-truncation.

(ii) For any family P of k-types over Y, the precomposition map

$$- \circ f : \left(\textstyle\prod_{(y:Y)} P(y)\right) \to \left(\textstyle\prod_{(x:X)} P(f(x))\right)$$

is an equivalence. This property is also called the ***dependent universal property*** *of the k-truncation.*

Proof. The fact that (ii) implies (i) is immediate, so we only have to prove the converse.

Suppose P is a family of k-truncated types over Y. Then we have a commuting square

$$\begin{tikzcd}
\Big(Y \to \sum_{(y:Y)} P(y)\Big) \arrow[r, "-\circ f"] \arrow[d, "\mathsf{pr}_1\circ -"'] & \Big(X \to \sum_{(y:Y)} P(y)\Big) \arrow[d, "\mathsf{pr}_1\circ -"] \\
\Big(Y \to Y\Big) \arrow[r, "-\circ f"'] & \Big(X \to Y\Big)
\end{tikzcd}$$

Since the total space $\sum_{(y:Y)} P(y)$ is again k-truncated by Exercise 10.3, it follows by the universal property of the k-truncation that the top map is an equivalence, and by the universal property the bottom map is an equivalence too. It follows from Corollary 21.5.5 that this square is a pullback square, so it induces equivalences on the fibers by Theorem 21.5.3. In particular we have a commuting square

$$\begin{tikzcd}
\Big(\prod_{(y:Y)} P(y)\Big) \arrow[r] \arrow[d] & \Big(\prod_{(x:X)} P(f(x))\Big) \arrow[d] \\
\mathsf{fib}_{(\mathsf{pr}_1\circ -)}(\mathsf{id}_Y) \arrow[r] & \mathsf{fib}_{(\mathsf{pr}_1\circ -)}(f)
\end{tikzcd}$$

in which the left and right maps are equivalences by Exercise 11.18, and the bottom map is an equivalence as we have just established. Therefore the top map is an equivalence, so we conclude that f satisfies the dependent universal property. □

Just as for pullbacks, pushouts, and the many other types characterized by a universal property, the k-truncation of a type is unique once it exists. We prove this in the following proposition and its corollary.

Proposition 30.1.4. *Consider a commuting triangle*

$$\begin{tikzcd}
& X \arrow[dl, "f"'] \arrow[dr, "f'"] & \\
Y \arrow[rr, "h"'] & & Y'
\end{tikzcd}$$

where Y and Y' are assumed to be k-types. If any two of the following three properties hold, so does the third:

(i) The map $f : X \to Y$ satisfies the universal property of the k-truncation of X.

(ii) The map $f' : X \to X'$ satisfies the universal property of the k-truncation of X.

(iii) The map h is an equivalence.

Proof. The claim follows by the 3-for-2 property of equivalences, since we have a commuting triangle

$$\begin{tikzcd}
Z^{Y'} \arrow[rr, "-\circ h"] \arrow[dr, "-\circ f'"'] & & Z^{Y} \arrow[dl, "-\circ f"] \\
& Z^X &
\end{tikzcd}$$

for any k-type Z. □

Corollary 30.1.5. *Consider two maps $f : X \to Y$ and $f' : X \to Y'$ into k-types Y and Y', and suppose that both f and f' satisfy the universal property of the k-truncation of X. Then the type of equivalences $e : Y \simeq Y'$ equipped with a homotopy witnessing that the triangle*

$$\begin{tikzcd} & X \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ Y \arrow[rr, "e"'] & & Y' \end{tikzcd}$$

commutes is contractible.

30.2 The construction of the $(k+1)$-truncation as a quotient

Definition 30.2.1. Consider a universe $\mathcal{U}$. We say that $\mathcal{U}$ **has k-truncations** if for every type $X : \mathcal{U}$ there is a map $f : X \to Y$ into a k-type $Y : \mathcal{U}$ that satisfies the universal property of the k-truncation of X.

Remark 30.2.2. Note that the universal property of k-truncations is formulated with respect to all k-types, and not only with respect to the k-types in $\mathcal{U}$.

We will use the following proposition to prove the universal property of the $(k+1)$-truncation. In fact, the converse of the following proposition also holds, and we prove it below in Theorem 30.2.6.

Proposition 30.2.3. *Consider a map $f : X \to Y$ into a $(k+1)$-type Y. If f is surjective, and its action on paths*

$$\mathsf{ap}_f : (x = x') \to (f(x) = f(x'))$$

satisfies the universal property of the k-truncation of $x = x'$, then f satisfies the universal property of the $(k+1)$-truncaton of X.

Proof. Consider a map $g : X \to Z$ into a $(k+1)$-type Z. Our goal is to show that g extends uniquely along f to a map $h : Y \to Z$. We claim that for any $y : Y$, the type of extensions

$$\begin{tikzcd} \mathsf{fib}_f(y) \arrow[d] \arrow[dr, "g \circ \mathsf{pr}_1"] & \\ \mathbf{1} \arrow[r, dotted] & Z \end{tikzcd}$$

is contractible. In other words, on each of the fibers of f, the map g is constant in a unique way. Since f is assumed to be surjective, it follows by Proposition 14.2.3 that it suffices to prove the above extension property for $y \equiv f(x)$, for each $x : X$. In other words, we have to show that the type

$$\textstyle\sum_{(z:Z)} \prod_{(x':X)} (f(x) = f(x')) \to (z = g(x'))$$

is contractible for each $x : X$.

Note that the type $z = g(x')$ is k-truncated, and that the map ap_f is assumed to satisfy the universal property of the k-truncation of $x = x'$. Therefore it is equivalent to show that the type

$$\textstyle\sum_{(z:Z)} \prod_{(x':X)} (x = x') \to (z = g(x'))$$

is contractible. This is immediate by the universal property of the identity type (Theorem 11.3.3), and the fact that $\sum_{(z:Z)} z = g(x')$ is contractible (Corollary 8.3.7).

It follows by Theorem 11.1.2 that the product

$$\prod_{(y:Y)}\sum_{(z:Z)}\prod_{(x:X)}(y = f(x)) \to (z = g(x))$$

is contractible. Since Π distributes over Σ by Theorem 11.2.1, we obtain that the type of functions $h : Y \to Z$ equipped with a homotopy $h \circ f \sim g$ is contractible. □

Before we show that any universe has k-truncations for arbitrary k, we prove a truncated version of the type theoretic Yoneda lemma under the assumption that $\mathcal{U}$ has k-truncations for a given k.

Lemma 30.2.4. *Suppose $\mathcal{U}$ is a universe that has k-truncations*

$$|-|_k : X \to \|X\|_k$$

for a given $k \geq -2$, and consider a family P of types over X. We make two claims:

(i) The evaluation function

$$\Big(\prod_{(y:X)}\|x = y\|_k \to \|P(y)\|_k\Big) \to \|P(x)\|_k$$

given by $h \mapsto h_x(|\mathsf{refl}_x|_k)$, is an equivalence.

(ii) If the total space of P is contractible, then the evaluation function

$$\Big(\prod_{(y:X)}\|x = y\|_k \simeq \|P(y)\|_k\Big) \to \|P(x)\|_k$$

given by $e \mapsto e_x(|\mathsf{refl}_x|_k)$, is an equivalence.

Proof. The first claim follows immediately by the universal property of the k-truncation of $x = y$ and the type theoretical Yoneda lemma (Theorem 11.3.3).

To prove the second claim, we first observe that the inclusion of equivalences into all maps induces an embedding that fits in a commuting triangle

$$\begin{tikzcd}
\Big(\prod_{(y:X)}\|x = y\|_k \simeq \|P(y)\|_k\Big) \arrow[rr, hook] \arrow[dr, "\mathsf{ev}_{|\mathsf{refl}_x|_k}"'] & & \Big(\prod_{(y:X)}\|x = y\|_k \to \|P(y)\|_k\Big) \arrow[dl, "\mathsf{ev}_{|\mathsf{refl}_x|_k}"] \\
& \|P(x)\|_k. &
\end{tikzcd}$$

The evaluation map on the right is an equivalence, and we have to show that if the total space $\sum_{(y:X)} P(y)$ is contractible, then the evaluation map on the left is an equivalence. We do this by showing that the top map is an equivalence.

To see this, note that we have a commuting diagram

$$\begin{tikzcd}
\Big(\prod_{(y:X)}(x = y) \simeq P(y)\Big) \arrow[r, hook] \arrow[d, "e \mapsto \lambda y.\, \|e_y\|_k"'] & \Big(\prod_{(y:X)}(x = y) \to P(y)\Big) \arrow[r, "\mathsf{ev\text{-}refl}"] \arrow[d, "h \mapsto \lambda y.\, \|h_y\|_k"'] & P(x) \arrow[d] \\
\Big(\prod_{(y:X)}\|x = y\|_k \simeq \|P(y)\|_k\Big) \arrow[r, hook] & \Big(\prod_{(y:X)}\|x = y\|_k \to \|P(y)\|_k\Big) \arrow[r, "\mathsf{ev}_{|\mathsf{refl}_x|_k}"'] & \|P(x)\|_k
\end{tikzcd}$$

In the top row of this diagram we have a concatenation of equivalences: the first map is an equivalence by the fundamental theorem of identity types, and the second map is an equivalence by Theorem 11.3.3. The second map in the bottom row is an equivalence by the first claim of this lemma. Therefore it follows that the vertical map in the middle satisfies the universal property of the k-truncation. Since the type at the bottom left is k-truncated, we obtain by the universal property of the k-truncation a section of the embedding in the bottom row, which proves the claim. □

Theorem 30.2.5. *Any univalent universe $\mathcal{U}$ that is closed under pushouts has k-truncations, for every truncation level k. We will write*

$$|-|_k : X \to \|X\|_k$$

for the k-truncation of X.

Proof. It is easy to see that the terminal projection $X \to \mathbf{1}$ is a (-2)-truncation, for any type X. Thus, any universe has (-2)-truncations.

We will proceed by induction on k. Our inductive hypothesis is that $\mathcal{U}$ has k-truncations, and our goal is to show that $\mathcal{U}$ has $(k+1)$-truncations. The idea of the construction is very similar to the construction of the set quotient by an equivalence relation. Consider the type-valued relation $R_k : X \to (X \to \mathcal{U})$ given by

$$R_k(x, x') :\equiv \|x = x'\|_k.$$

Analogous to the definition of set quotients, we define

$$\|X\|_{k+1} :\equiv \mathrm{im}(R_k),$$

which comes equipped with a surjective map $q : X \to \|X\|_{k+1}$. Note that the image of $R : X \to (X \to \mathcal{U})$ is (essentially) small by Theorem 27.6.11. To see that $q : X \to \|X\|_{k+1}$ satisfies the universal property of the $(k+1)$-truncation of X, we apply Proposition 30.2.3. Therefore it remains to show that the action on paths

$$\mathsf{ap}_q : (x = x') \to (q(x) = q(x'))$$

satisfies the universal property of the k-truncation of $x = x'$. Since q is the surjective map in the image factorization of R_k, it is equivalent to show that the action on paths

$$\mathsf{ap}_R : (x = x') \to (R_k(x) = R_k(x'))$$

satisfies the universal property of the k-truncation of $x = x'$. Note that we have a commuting triangle

$$\begin{tikzcd} & (x = x') \arrow[dl, "\mathsf{ap}_{R_k}"'] \arrow[dr] & \\ \Big(R_k(x) = R_k(x')\Big) \arrow[rr, "\simeq"'] & & \Big(\prod_{(y:X)} \|x = y\|_k \simeq \|x' = y\|_k\Big), \end{tikzcd}$$

where the bottom map is an equivalence by function extensionality and the univalence axiom. Therefore it suffices to show that the map on the right of this triangle, which is the unique map that sends refl_x to the family of identity equivalences, satisfies the universal property of the k-truncation of $x = x'$.

This map fits in a commuting square

$$\begin{tikzcd}
(x = x') \arrow[r, "|-|_k"] \arrow[d] & \|x = x'\|_k \arrow[d, "\|\mathsf{inv}\|_k"] \\
\left(\prod_{(y:X)} \|x = y\|_k \simeq \|x' = y\|_k\right) \arrow[r, "\mathsf{ev}_{|\mathsf{refl}_x|_k}"'] & \|x' = x\|_k.
\end{tikzcd}$$

The map on the right is an equivalence because $\mathsf{inv} : (x = x') \to (x' = x)$ is an equivalence. The bottom map is an equivalence by Lemma 30.2.4. The top map satisfies the universal property of the k-truncation of $x = x'$, hence so does the map on the left, which completes the proof. □

Theorem 30.2.6. *Consider a map $f : X \to Y$ into a $(k+1)$-truncated type Y. Then the following are equivalent:*

(i) The map f satisfies the universal property of the $(k+1)$-truncation of X.

(ii) The map f is surjective, and for each $x, x' : X$ the map

$$\mathsf{ap}_f : (x = x') \to (f(x) = f(x'))$$

satisfies the universal property of the k-truncation of $x = x'$.

Proof. The fact that (ii) implies (i) was established in Proposition 30.2.3, so it suffices to show that (i) implies (ii).

Suppose first that the map f satisfies the universal property of the $(k+1)$-truncation, and let $x : X$. Recall from Exercise 12.1 that the universe of k-truncated types is itself $(k+1)$-truncated. Therefore it follows that the map $x' \mapsto \|x = x'\|_k$ has a unique extension

$$\begin{tikzcd}
X \arrow[d, "f"'] \arrow[dr, "x' \mapsto \|x = x'\|_k"] & \\
Y \arrow[r, dotted, "P"'] & \mathcal{U}_{\leq k}.
\end{tikzcd}$$

In other words, we obtain a unique family P of k-types over Y equipped with equivalences

$$e_{x'} : P(f(x')) \simeq \|x = x'\|_k$$

indexed by $x' : X$. In particular, P comes equipped with a point $p_0 : P(f(x))$ such that $e_x(p_0) = |\mathsf{refl}_x|_k$. Hence we obtain a family of maps

$$\prod_{(y:Y)} (f(x) = y) \to P(y).$$

We claim that this is a family of equivalences. By the fundamental theorem of identity types, Theorem 9.2.2, it suffices to show that the total space

$$\sum_{(y:Y)} P(y)$$

is contractible. We have $(f(x), p_0)$ at the center of contraction, so we have to construct a contraction

$$\prod_{(y:Y)} \prod_{(p:P(y))} (f(x), p_0) = (y, p).$$

Now we observe that the type $\sum_{(y:Y)} P(y)$ is $(k+1)$-truncated, using the fact that any Σ-type of a family of k-types over a $(k+1)$-type is again $(k+1)$-truncated (Exercise 10.3). It follows that the type $(f(x), p_0) = (y, p)$ is k-truncated for each $y : Y$ and each $p : P(y)$. Now we use the dependent universal property of the k-truncation of X, which was proven in Proposition 30.1.3, so it suffices to show that

$$\prod_{(x':X)} \prod_{(p:P(f(x')))} (f(x), p_0) = (f(x'), p).$$

Since we have an equivalence $e_{x'} : P(f(x')) \simeq \|x = x'\|_k$ for each $x' : X$, it is equivalent to show that

$$\prod_{(x':X)} \prod_{(p:\|x=x'\|_k)} (f(x), p_0) = (f(x'), e_{x'}^{-1}(p)).$$

Again, we use that the type of paths $(f(x), p_0) = (f(x'), e_{x'}^{-1}(p))$ is a k-type, so we use Proposition 30.1.3 to conclude that it suffices to show that

$$\prod_{(x':X)} \prod_{(p:x=x')} (f(x), p_0) = (f(x'), e_{x'}^{-1}(p)).$$

This is immediate by path induction on $p : x = x'$. This proves the claim that the canonical map

$$h_y : (f(x) = y) \to P(y)$$

is an equivalence for each $y : Y$. Now observe that we have a commuting triangle

$$\begin{tikzcd} & (x = x') \arrow[dl, "\mathsf{ap}_f"'] \arrow[dr, "|-|_k"] & \\ (f(x) = f(x')) \arrow[rr, "h_{f(x')}"'] & & \|x = x'\|_k \end{tikzcd}$$

for each $x' : X$. Therefore it follows that $\mathsf{ap}_f : (x = x') \to (f(x) = f(x'))$ satisfies the universal property of the k-truncation of $x = x'$. $\square$

Corollary 30.2.7. *For any $x, y : X$, there is an equivalence*

$$(|x|_{k+1} = |y|_{k+1}) \simeq \|x = y\|_k.$$

30.3 The truncations as recursive higher inductive types

Recall from Theorem 10.3.6 that a map $f : A \to B$ is $(k+1)$-truncated if and only if the action on paths

$$\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$$

is a k-truncated map, for each $x, y : A$. Moreover, in Exercise 21.2 we established that the fibers of the diagonal map $\delta_f : A \to A \times_B A$ are equivalent to the fibers of the maps ap_f, so it is also the case that f is $(k+1)$-truncated if and only if the diagonal δ_f is k-truncated.

In the following theorem, we add yet another equivalent characterization to the truncatedness of a map. We will use this theorem in two ways. First, a simple corollary gives a useful characterization of k-truncated types. Second, we will use this theorem to derive an elimination principle of the $(k+1)$-sphere that can be applied to families of k-types

Theorem 30.3.1. *Consider a map $f : A \to B$. Then the following are equivalent:*

(i) The map f is k-truncated.

(ii) The commuting square

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\lambda x.\, \mathsf{const}_x"'] & B \arrow[d, "\lambda y.\, \mathsf{const}_y"] \\ A^{\mathbf{S}^{k+1}} \arrow[r, "f^{\mathbf{S}^{k+1}}"'] & B^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square.

Proof. We prove the claim by induction on $k \geq -2$. The base case is clear, because the map $A^{\mathbf{S}^{-1}} \to B^{\mathbf{S}^{-1}}$ is a map between contractible types, hence an equivalence. Therefore the square

$$\begin{tikzcd} A \arrow[r] \arrow[d] & B \arrow[d] \\ A^{\mathbf{S}^{-1}} \arrow[r] & B^{\mathbf{S}^{-1}} \end{tikzcd}$$

is a pullback square if and only if $A \to B$ is an equivalence.

For the inductive step, assume that for any map $g : X \to Y$, the map g is k-truncated if and only if the square

$$\begin{tikzcd} X \arrow[r] \arrow[d] & Y \arrow[d] \\ X^{\mathbf{S}^{k+1}} \arrow[r] & Y^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square, and consider a map $f : A \to B$. Then f is $(k+1)$-truncated if and only if $\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$ is k-truncated for each $x, y : A$. By the inductive hypothesis this happens if and only if the square

$$\begin{tikzcd} (x = y) \arrow[r] \arrow[d] & (f(x) = f(y)) \arrow[d] \\ (x = y)^{\mathbf{S}^{k+1}} \arrow[r] & (f(x) = f(y))^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square, for each $x, y : A$. Now we observe that this is the case if and only if the square on the left in the diagram

$$\begin{tikzcd} \sum_{(x,y:A)} x = y \arrow[r] \arrow[d] & \sum_{(x,y:A)} (f(x) = f(y)) \arrow[r] \arrow[d] & \sum_{(x,y:B)} x = y \arrow[d] \\ \sum_{(x,y:A)} (x = y)^{\mathbf{S}^{k+1}} \arrow[r] & \sum_{(x,y:A)} (f(x) = f(y))^{\mathbf{S}^{k+1}} \arrow[r] & \sum_{(x,y:B)} (x = y)^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square. The square on the right is a pullback square, so the square on the left is a pullback if and only if the outer rectangle is a pullback. By the universal property of $\mathbf{S}^{k+2}$ it follows that the outer rectangle is a pullback if and only if the square

$$\begin{tikzcd} A \arrow[r] \arrow[d] & B \arrow[d] \\ A^{\mathbf{S}^{k+2}} \arrow[r] & B^{\mathbf{S}^{k+2}} \end{tikzcd}$$

is a pullback. $\square$

Theorem 30.3.2. *Consider a type A. Then the following are equivalent:*

(i) The type A is k-truncated.

(ii) The map
$$\lambda x.\,\mathsf{const}_x : A \to (\mathbf{S}^{k+1} \to A)$$
is an equivalence.

Proof. We prove the claim by induction on $k \geq -2$. The base case is clear, because the map $A^{\mathbf{S}^{-1}}$ is contractible.

For the inductive step, assume that any type X is k-truncated if and only if the map
$$\lambda x.\,\mathsf{const}_x : X \to (\mathbf{S}^{k+1} \to X)$$
is an equivalence. Then A is $(k+1)$-truncated if and only if its identity types $x = y$ are k-truncated, for each $x, y : A$. By the inductive hypothesis this happens if and only if
$$(x = y) \to (\mathbf{S}^{k+1} \to (x = y))$$
is a family of equivalences indexed by $x, y : A$. This is a family of equivalences if and only if the induced map on total spaces
$$\Big(\textstyle\sum_{(x,y:A)} x = y\Big) \to \Big(\textstyle\sum_{(x,y:A)} (x = y)^{\mathbf{S}^{k+1}}\Big)$$
is an equivalence. Note that we have a commuting square
$$\begin{tikzcd}
A \arrow[r] \arrow[d] & A^{\mathbf{S}^{k+2}} \arrow[d] \\
\Big(\sum_{(x,y:A)} x = y\Big) \arrow[r] & \Big(\sum_{(x,y:A)} (x = y)^{\mathbf{S}^{k+1}}\Big)
\end{tikzcd}$$
in which both vertical maps are equivalences. Therefore the top map is an equivalence if and only if the bottom map is an equivalence, which completes the proof. $\square$

Proof. Immediate from the fact that A is k-truncated if and only if the map $A \to \mathbf{1}$ is k-truncated. $\square$

Definition 30.3.3. Consider a type X. A k**-truncation** of X consist of a k-type Y, and a map $f : X \to Y$ satisfying the **universal property of k-truncation**, that for every k-type Z the precomposition map
$$- \circ f : (Y \to Z) \to (X \to Z)$$
is an equivalence.

We define $\|X\|_k$ by the 'hubs-and-spokes' method, as a higher inductive type. The idea is to force any map $\mathbf{S}^k X \to \|X\|_k$ to be homotopic to a constant function by including enough points (the hubs) for the values of these constant functions, and enough paths (the spokes) for the homotopies to these constant functions.

Definition 30.3.4. For any type X we define a type $\|X\|_k$ as a higher inductive type, with constructors

$$\begin{aligned} \eta &: X \to \|X\|_k. \\ \mathsf{hub} &: (\mathbf{S}^{k+1} \to \|X\|_k) \to \|X\|_k \\ \mathsf{spoke} &: \prod_{(f:\mathbf{S}^{k+1}\to\|X\|_k)} \prod_{(t:\mathbf{S}^{k+1})} f(t) = \mathsf{hub}(f). \end{aligned}$$

Remark 30.3.5. The induction principle for $\|X\|_k$ asserts that for any family P of types over $\|X\|_k$, if we have a dependent function $\alpha : \prod_{(x:X)} P(\eta(x))$ and a dependent function

$$\beta : \prod_{(f:\mathbf{S}^{k+1}\to\|X\|_k)} \Big(\prod_{(t:\mathbf{S}^{k+1})} P(f(t))\Big) \to P(\mathsf{hub}(f))$$

equipped with an identification

$$\gamma(f,g,t) : \mathsf{tr}_P(\mathsf{spoke}(f,t), g(t)) = \beta(f,g),$$

for every $f : \mathbf{S}^{k+1} \to \|X\|_k$, $g : \prod_{(t:\mathbf{S}^{k+1})} P(f(t))$, and every $t : \mathbf{S}^{k+1}$, then we obtain a dependent function

$$h : \prod_{(x:\|X\|_k)} P(\eta(x))$$

equipped with an identification $H(x) : h(\eta(x)) = \alpha(x)$ for any $x : X$.

Proposition 30.3.6. *For any type X, the type $\|X\|_k$ is k-truncated.*

Proof. By Theorem 30.3.2 it suffices to show that the map

$$\delta :\equiv \lambda x.\, \mathsf{const}_x : \|X\|_k \to (\mathbf{S}^{k+1} \to \|X\|_k)$$

is an equivalence. Note that the inverse of this map is simply the map

$$\mathsf{hub} : (\mathbf{S}^{k+1} \to \|X\|_k) \to \|X\|_k,$$

which is a section of δ by the homotopy spoke. Therefore it remains to show that

$$\mathsf{hub}(\mathsf{const}_x) = x.$$

for every $x : \|X\|_k$. Note that $\mathsf{spoke}(\mathsf{const}_x, \mathsf{hub}(\mathsf{const}_x))^{-1}$ is such an identification. □

Recall that the $(k+1)$-sphere is k-connected in the following sense.

Lemma 30.3.7. *For any family P of k-types over $\mathbf{S}^{k+1}$, the evaluation map at the base point*

$$\mathsf{ev}_* : \Big(\prod_{(t:\mathbf{S}^{k+1})} P(t)\Big) \to P(*)$$

is an equivalence.

Theorem 30.3.8. *For any family P of k-types over $\|X\|_k$, the function*

$$- \circ \eta : \Big(\prod_{(x:\|X\|_k)} P(x)\Big) \to \Big(\prod_{(x:X)} P(\eta(x))\Big)$$

is an equivalence.

Proof. We first show that for any family P of k-types over $\|X\|_k$, the function

$$-\circ\eta : \Big(\prod_{(x:\|X\|_k)} P(x)\Big) \to \Big(\prod_{(x:X)} P(\eta(x))\Big)$$

has a section. To see this, we apply the induction principle of $\|X\|_k$. For any function $\alpha : \prod_{(x:X)} P(\eta(x))$ we need to construct a function $h : \prod_{(x:\|X\|_k)} P(x)$ such that $h \circ \eta \sim \alpha$, so it suffices to show that the k-truncatedness of the types in the family P imply the existence of the terms β and η of the induction principle of $\|X\|_k$. In other words, we need to show that for every $f : \mathbf{S}^{k+1} \to \|X\|_k$ and every $g : \prod_{(t:\mathbf{S}^{k+1})} P(f(t))$ there are

$$\beta(f,g) : P(\mathsf{hub}(f))$$
$$\gamma(f,g) : \prod_{(t:\mathbf{S}^{k+1})} \mathsf{tr}_P(\mathsf{spoke}(f,t), g(t)) = \beta(f,g).$$

Since we have already shown that $\|X\|_k$ is k-truncated, it suffices to show the above for $f :\equiv \mathsf{const}_x$, for any $x : \|X\|_k$. Now the type of g is just the function type $\mathbf{S}^{k+1} \to P(x)$, so by the truncatedness of $P(x)$ it suffices to construct

$$\beta(\mathsf{const}_x, \mathsf{const}_y) : P(\mathsf{hub}(\mathsf{const}_x))$$
$$\gamma(\mathsf{const}_x, \mathsf{const}_y) : \prod_{(t:\mathbf{S}^{k+1})} \mathsf{tr}_P(\mathsf{spoke}(\mathsf{const}_x, t), y) = \beta(\mathsf{const}_x, \mathsf{const}_y)$$

for any $x : X$ and $y : P(x)$. Now we simply define

$$\beta(\mathsf{const}_x, \mathsf{const}_y) :\equiv \mathsf{tr}_P(\mathsf{spoke}(\mathsf{const}_x, *), y).$$

Then it remains to construct an identification

$$\mathsf{tr}_P(\mathsf{spoke}(\mathsf{const}_x, t), y) = \mathsf{tr}_P(\mathsf{spoke}(\mathsf{const}_x, *), y)$$

for any $t : \mathbf{S}^{k+1}$, but this follows at once from Lemma 30.3.7, because the identity types of a k-truncated type is again k-truncated. This completes the proof that the precomposition function

$$-\circ\eta : \Big(\prod_{(x:\|X\|_k)} P(x)\Big) \to \Big(\prod_{(x:X)} P(\eta(x))\Big)$$

has a section s for every family P of k-types over $\|X\|_k$.

To show that it is an equivalence, we have to show that s is also a retraction of the precomposition function $-\circ\eta$, i.e., we have to show that

$$s(h\circ\eta) = h$$

for any $h : \prod_{(x:\|X\|_k)} P(x)$. By function extensionality, it is equivalent to show that

$$\prod_{(x:\|X\|_k)} s(h\circ\eta)(x) = h(x).$$

Now we observe that the type $s(h\circ\eta)(x) = h(x)$ is a k-type, and therefore we already know that the function

$$-\circ\eta : \Big(\prod_{(x:\|X\|_k)} s(h\circ\eta)(x) = h(x)\Big) \to \Big(\prod_{(x:X)} s(h\circ\eta)(\eta(x)) = h(\eta(x))\Big)$$

has a section. In other words, it suffices to construct a dependent function of type

$$\prod_{(x:X)} s(h\circ\eta)(\eta(x)) = h(\eta(x)).$$

Here we simply use that s is a section $-\circ\eta$, and we are done. □

Corollary 30.3.9. *For any type X, the map $\eta : X \to \|X\|_k$ satisfies the universal property of k-truncation.*

30.4 Theorems not to forget

Theorem 30.4.1. *Consider a type X and a family P of $(k+n)$-truncated types over $\|X\|_k$. Then the precomposition map*

$$-\circ\eta : \Big(\prod_{(y:\|X\|_k)} P(y)\Big) \to \Big(\prod_{(x:X)} P(\eta(x))\Big)$$

is $(n-2)$-truncated.

Exercises

30.1 Consider an equivalence relation $R : A \to (A \to \mathsf{Prop})$. Show that the map $|-|_0 \circ \mathsf{inl} : A \to \|A \sqcup^R A\|_0$ satisfies the universal property of the quotient A/R, where $A \sqcup^R A$ is the canonical pushout

$$\begin{tikzcd} \sum_{(x,y:A)} R(x,y) \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A \arrow[d, "\mathsf{inr}"] \\ A \arrow[r, "\mathsf{inl}"'] & A \sqcup^R A. \end{tikzcd}$$

30.2 Consider the trivial relation $\mathbf{1} :\equiv \lambda x.\, \lambda y.\, \mathbf{1} : A \to (A \to \mathsf{Prop})$. Show that the set quotient $A/\mathbf{1}$ is a proposition satisfying the universal property of the propositional truncation.

30.3 Show that the type of pointed 2-element sets

$$\sum_{(X:\mathcal{U}_\mathbf{2})} X$$

is contractible.

30.4 Define the type $\mathbb{F}$ of finite sets by

$$\mathbb{F} :\equiv \mathrm{im}(\mathsf{Fin}),$$

where $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$ is defined in Definition 6.4.1.

(a) Show that $\mathbb{F} \simeq \sum_{(n:\mathbb{N})} \mathcal{U}_{\mathsf{Fin}(n)}$.
(b) Show that $\mathbb{F}$ is closed under Σ and Π.

30.5 (a) A type Y is called **k-separated** if for every type X the map

$$(\|X\|_k \to Y) \to (X \to Y)$$

is an embedding. Show that Y is k-separated if and only if it is $(k+1)$-truncated.

(b) A type Y is called **n-fold k-separated** if for every type X the map

$$(\|X\|_k \to Y) \to (X \to Y)$$

is $(n-2)$-truncated. Show that Y is n-fold k-separated if and only if it is $(k+n)$-truncated.

30.6 Consider a map $f : A \to B$. Show that the square

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\lambda x.\, \mathsf{const}_x"'] & B \arrow[d, "\lambda y.\, \mathsf{const}_y"] \\ A^{\mathbf{S}^{k+1}} \arrow[r, "f^{\mathbf{S}^{k+1}}"'] & B^{\mathbf{S}^{k+1}} \end{tikzcd}$$

is a pullback square if and only if its gap map has a section.

30.7 Consider a map $f : X \to Y$ into a k-truncated type Y. Show that the following are equivalent:

(i) For any family P of k-types over Y, the precomposition map

$$- \circ f : \left(\prod_{(y:Y)} P(y)\right) \to \left(\prod_{(x:X)} P(f(x))\right)$$

is an equivalence.

(ii) For any family P of k-types over Y, the precomposition map

$$- \circ f : \left(\prod_{(y:Y)} P(y)\right) \to \left(\prod_{(x:X)} P(f(x))\right)$$

has a section.

30.8 Show that for each type X, the map

$$\|X\|_{k+1} \to \mathcal{U}^X$$

given by $y \mapsto \lambda x. (y = \eta'(x))$ is an embedding.

31 Connected types and maps

In this section we introduce the concept of k-connected types and maps. We define k-connected types to be types with contractible k-truncation, and a k-connected map is just a map of which the fibers are k-connected. The idea is that a type is k-connected if and only if its homotopy groups $\pi_i(X)$ are trivial for all $i \leq k$.

One of the main theorems in this section is a characterization of k-connected maps in terms of their action on homotopy groups: A map $f : X \to Y$ is k-connected if and only if it induces isomorphisms

$$\pi_i(f, x) : \pi_i(X, x) \to \pi_i(Y, f(x))$$

of homotopy groups, for each $i \leq k$ and each $x : X$, and a *surjective* group homomorphism

$$\pi_{k+1}(f, x) : \pi_{k+1}(X, x) \to \pi_{k+1}(Y, f(x))$$

on the $(k+1)$-st homotopy group, for each $x : X$. If one drops the condition that f induces a surjective group homomorphism on the $(k+1)$-st homotopy group, then the map is only a k-equivalence, i.e., a map of which $\|f\|_k$ is an equivalence. We see from the above characterization that any k-connected map is a k-equivalence, and also that any $(k+1)$-equivalence is a k-connected map. Nevertheless, the difference between the classes of k-equivalences and k-connected maps is somewhat subtle.

We will study k-equivalences and k-connected maps synchronously, because understanding the subtle differences between the results about either of them will increase the understanding of both classes of maps. For instance, we will show that the k-connected maps enjoy a dependent elimination property, while the k-equivalences only satisfy a non-dependent elimination property. We will see that the k-equivalences satisfy the 3-for-2 property, while one of the cases of the 3-for-2 property fails for k-connected maps.

The k-connected maps can be characterized as the class of maps that is left orthogonal to the class of k-truncated maps, where a map $f : A \to B$ is said to be left orthogonal to a map

$g : X \to Y$ if the type of diagonal fillers of any commuting square of the form

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & X \arrow[d, "g"] \\ B \arrow[r] \arrow[ur, dotted] & Y \end{tikzcd}$$

is contractible. Similarly, the class of k-equivalences is the class of maps that is left orthogonal to any map between k-truncated types. However, this result is not entirely sharp, because there are more maps that the k-equivalences are left orthogonal to. It turns out that a map is a k-equivalence if and only if it is left orthogonal to any map $g : X \to Y$ for which the naturality square

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[d, "\eta"'] & Y \arrow[d] \\ \|X\|_k \arrow[r, "\|g\|_k"'] & \|Y\|_k \end{tikzcd}$$

is a pullback square. Such maps are called k-étale, and they induce isomorphisms

$$\pi_i(g, x) : \pi_i(X, x) \to \pi_i(Y, g(x))$$

on homotopy groups for $i > k$.

In the final part of this section we will use the results about k-equivalences to show that the n-sphere is $(n-1)$-connected, for each $n : \mathbb{N}$, and that the join $A * B$ is $(k+l+2)$-connected if A is k-connected and B is l-connected.

31.1 Connected types

Definition 31.1.1. A type X is said to be ***k*-connected** if its k-truncation $\|X\|_k$ is contractible. We define

$$\mathsf{is\text{-}conn}_k(X) :\equiv \mathsf{is\text{-}contr}\|X\|_k.$$

Remark 31.1.2. Since the (-2)-truncation of any type is just $\mathbf{1}$, it follows that every type is (-2)-connected. Furthermore, since any proposition is contractible as soon as it comes equipped with a term, it follows that any type is (-1)-connected as soon as it is inhabited.

In Theorem 31.1.4 below, we will see that a type X is 0-connected if and only if it is inhabited and every two points are connected by an unspecified path. In this sense 0-connected types are also called **path connected**, or just **connected**. Thus, it is immediate that the circle is an example of a connected type.

Similarly, in the case where $k \equiv 0$ the theorem states that a type X is 1-conneced if and only if it is inhabited and for every $x, y : X$ the identity type $x = y$ is path connected. In other words, a type is **simply connected** if it is 1-connected! The 2-sphere is an example of a simply connected type. This fact is shown in Corollary 31.4.4 below, where we will show more generally that the n-sphere is $(n-1)$-connected, for each $n : \mathbb{N}$.

Lemma 31.1.3. *If a type is $(k+1)$-connected, then it is also k-connected.*

Proof. This follows from the fact that $\|\|X\|_{k+1}\|_k \simeq \|X\|_k$. Indeed, if $\|X\|_{k+1}$ is contractible, then its k-truncation is also contractible, so it follows that $\|X\|_k$ is contractible. □

For the following theorem, recall that a type X is said to be inhabited if it comes equipped with a term $\|X\|_{-1}$.

Theorem 31.1.4. *Consider a type X. Then the following are equivalent:*

(i) The type X is $(k+1)$-connected.

(ii) The type X is inhabited, and the type $x = y$ is k-connected for each $x, y : X$.

Proof. Suppose first that X is $(k+1)$-connected. It is immediate that X is inhabited in this case. Moreover, since we have equivalences

$$(\eta(x) = \eta(y)) \simeq \|x = y\|_k$$

for each $x, y : X$, it follows from the assumption that $\|X\|_{k+1}$ is contractible that the type $\|x = y\|_k$ is equivalent to a contractible type. This proves that (i) implies (ii).

To see that (ii) implies (i), suppose that X is inhabited and that its identity types are k-connected. Our goal is to construct a term of type

$$\mathsf{is\text{-}contr}\|X\|_{k+1},$$

which is a proposition, so we may eliminate the assumption that X is inhabited and assume to have $x : X$. Now we simply take $\eta(x)$ for the center of contraction of $\|X\|_{k+1}$. To construct the contraction, note that by the dependent universal property of $(k+1)$-truncation we have an equivalence

$$\Big(\prod_{(y:\|X\|_{k+1})} \eta(x) = y\Big) \simeq \Big(\prod_{(y:X)} \eta(x) = \eta(y)\Big).$$

Therefore it suffices to construct an identification $\eta(x) = \eta(y)$ for every $y : X$. However, this type is contractible, since it is equivalent to the contractible type $\|x = y\|_k$. This completes the proof of (ii) implies (i). ☐

In the case where $k \geq -1$ we can improve Theorem 31.1.4 and characterize a high degree of connectedness entirely in terms of the triviality of homotopy groups. This is what connectedness is all about.

Theorem 31.1.5. *Consider a type X, and suppose that $k \geq 0$. Then the following are equivalent:*

(i) The type X is k-connected.

(ii) The type X is connected, and for every $x : X$ the loop space

$$\Omega(X, x)$$

is $(k-1)$-connected.

(iii) For each $i \leq k$ and each $x : X$, the i-th homotopy group $\pi_i(X, x)$ is trivial.

Proof. If X is k-connected for $k \geq 0$, then it is certainly connected, and $\Omega(X, x)$ is $(k-1)$-connected by Theorem 31.1.4. Thus, the fact that (i) implies (ii) is immediate.

To see that (ii) implies (i), note that if X is connected and its loop spaces are $(k-1)$-connected, then all its identity types are $(k-1)$-connected, since we have

$$\begin{aligned}
\prod_{(x,y:X)} \mathsf{is\text{-}contr}(\|x = y\|_{k-1}) &\simeq \prod_{(x,y:X)} \|x = y\|_{-1} \to \mathsf{is\text{-}contr}(\|x = y\|_{k-1}) \\
&\simeq \prod_{(x,y:X)} (x = y) \to \mathsf{is\text{-}contr}(\|x = y\|_{k-1}) \\
&\simeq \prod_{(x:X)} \mathsf{is\text{-}contr}(\|x = x\|_{k-1}).
\end{aligned}$$

In the first step of this calculation we use that X is connected, so $\|x = y\|_{-1}$ is contractible; then we use that $\mathsf{is\text{-}contr}(\|x = y\|_{k-1})$ is a proposition; and finally we use the universal property of identity types to arrive at our assumption that the loop spaces of X are $(k-1)$-connected. Since we have shown that the identity types are $(k-1)$-connected, it follows by Theorem 31.1.4 that X is k-connected, which concludes the proof that (ii) implies (i).

It is easy to see by induction on $k \geq 0$ that (ii) holds if and only if (iii) holds, since we have

$$\pi_{i+1}(X, x) = \pi_i(\Omega(X, x)). \qquad \square$$

Remark 31.1.6. If X is assumed to be a pointed type in Theorem 31.1.5, then conditions (ii) and (iii) only have to be checked at the base point.

31.2 k-Equivalences and k-connected maps

We now study two classes of maps that differ only slightly: the k-equivalences and the k-connected maps.

Definition 31.2.1.

(i) A map $f : X \to Y$ is said to be ***k*-connected** if its fibers are k-connected. We will write

$$\mathsf{is\text{-}conn}_k(f) :\equiv \prod_{(y:Y)} \mathsf{is\text{-}conn}_k(\mathsf{fib}_f(y)).$$

(ii) A map $f : X \to Y$ is said to be a ***k*-equivalence** if

$$\|f\|_k : \|X\|_k \to \|Y\|_k$$

is an equivalence. We will write

$$\mathsf{is\text{-}equiv}_k(f) :\equiv \mathsf{is\text{-}equiv}(\|f\|_k).$$

Example 31.2.2. Any equivalence is a k-connected map, as well as a k-equivalence. Moreover, for any k-connected type X the map $\mathsf{const}_\star : X \to \mathbf{1}$ is k-connected. It is also immediate that *any* map between k-connected types is a k-equivalence.

Example 31.2.3. A (-1)-connected map is a map $f : X \to Y$ for which the propositionally truncated fibers

$$\|\mathsf{fib}_f(y)\|_{-1}$$

are contractible. Since propositions are contractible as soon as they are inhabited, we see that a map is (-1)-connected if and only if it is surjective.

A (-1)-equivalence, on the other hand, is just a map $f : X \to Y$ that induces an equivalence $\|X\|_{-1} \simeq \|Y\|_{-1}$. The map $\mathsf{const}_{1_{\mathbf{2}}} : \mathbf{1} \to \mathbf{2}$ is an example of such a map, showing that (-1)-equivalences don't need to be surjective.

However, it is the case that every surjective map $f : X \to Y$ is in fact (-1)-equivalence. To see this, we need to show that

$$\|Y\|_{-1} \to \|X\|_{-1}.$$

Such a map is constructed by the universal property of (-1)-truncation. Thus, it suffices to construct a function $Y \to \|X\|_{-1}$. Since we have assumed that f is surjective, we have for every $y : Y$ a term

$$s(y) : \|\mathsf{fib}_f(y)\|_{-1}.$$

Thus, we define a function $Y \to \|X\|_{-1}$ by

$$y \mapsto \|\mathsf{pr}_1\|_{-1}(s(y)).$$

This concludes the proof that f is a (-1)-equivalence, since we have shown that $\|X\|_{-1} \leftrightarrow \|Y\|_{-1}$.

Remark 31.2.4. An immediate difference between the classes of k-equivalences and k-connected maps is that the k-connected maps are stable under base change, while the k-equivalences are not. By this, we mean that for any pullback square

$$\begin{tikzcd} E' \arrow[r, "g"] \arrow[d, "p'"'] & E \arrow[d, "p"] \\ B' \arrow[r, "f"'] & B, \end{tikzcd}$$

if the map p is k-connected, then the map p' is also k-connected. In such a pullback diagram, the map p' is sometimes called the **base change** of p along f. By Theorem 21.5.3 we have an equivalence

$$\mathsf{fib}_{p'}(b') \simeq \mathsf{fib}_p(f(b'))$$

for any $b' : B'$, so it is indeed the case that if the fibers of p are k-connected, then so are the fibers of p'.

An example showing that the k-equivalences are not stable under base change is given by the pullback square

$$\begin{tikzcd} \Omega(\mathbf{S}^{k+1}) \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \mathbf{1} \arrow[r] & \mathbf{S}^{k+1} \end{tikzcd}$$

We will show in Corollary 31.4.4 that the $(k+1)$-sphere is k-connected, so the map $\mathbf{1} \to \mathbf{S}^{k+1}$ is a k-equivalence. However, its loop space is only $(k-1)$-connected, and indeed we will show in **??** that $\pi_{k+1}(\mathbf{S}^{k+1}) = \mathbb{Z}$ for $k \geq 0$, showing that $\Omega(\mathbf{S}^{k+1})$ is *not* k-connected. Thus, the map $\Omega(\mathbf{S}^{k+1}) \to \mathbf{1}$ is not a k-equivalence.

Elimination properties

We will show that a map $f : X \to Y$ is a k-equivalence if and only if the precomposition function

$$- \circ f : (Y \to Z) \to (X \to Z)$$

is an equivalence for every k-type Z. On the other hand, we will show that f is k-connected if and only if the precomposition function

$$- \circ f : \left(\prod_{(y:Y)} P(y)\right) \to \left(\prod_{(x:X)} P(f(x))\right)$$

is an equivalence for every family P of k-types over Y. In other words, the k-connected maps satisfy a *dependent* unique elimination property, while the k-equivalences only satisfy a *non-dependent* unique elimination property.

Theorem 31.2.5. *Consider a function $f : X \to Y$. Then the following are equivalent*

(i) The map f is a k-equivalence.

(ii) *For every k-type Z, the precomposition function*

$$-\circ f : (Y \to Z) \to (X \to Z)$$

is an equivalence.

Theorem 31.2.6. *Let $f : X \to Y$ be a map. The following are equivalent:*

(i) *The map f is k-connected.*

(ii) *For every family P of k-truncated types over Y, the precomposition map*

$$-\circ f : \Big(\prod_{(y:Y)} P(y)\Big) \to \Big(\prod_{(x:X)} P(f(x))\Big)$$

is an equivalence.

Proof. Suppose f is k-connected and let P be a family of k-types over Y. Now we may consider the following commuting diagram

$$\begin{tikzcd}
& \prod_{(y:Y)} P(y) \arrow[rr, "-\circ f"] \arrow[dl] & & \prod_{(x:X)} P(f(x)) \\
\prod_{(y:Y)} \|\mathsf{fib}_f(y)\|_k \to P(y) \arrow[dr] & & & \prod_{(x:X)} \prod_{(y:Y)} \prod_{(p:f(x)=y)} P(y) \arrow[ul] \\
& \prod_{(y:Y)} \mathsf{fib}_f(y) \to P(y) \arrow[r] & \prod_{(y:Y)} \prod_{(x:X)} \prod_{(p:f(x)=y)} P(y) \arrow[ur] &
\end{tikzcd}$$

which commutes by $\mathsf{refl\text{-}htpy}$. In this diagram, the five maps going around counter clockwise are all equivalences for obvious reasons, so it follows that the top map is an equivalence.

Now suppose that f satisfies the dependent elimination property stated in (ii). In order to construct a center of contraction of $\|\mathsf{fib}_f(y)\|_k$ for every $y : Y$, we use the dependent elimination property with respect to the family P given by $P(y) :\equiv \|\mathsf{fib}_f(y)\|_k$. □

Corollary 31.2.7. *For any type X, the unit $\eta : X \to \|X\|_k$ of the k-truncation is a k-connected map.*

The inclusions

We will prove the following implications

$$\mathsf{is\text{-}equiv}_{k+1}(f) \xrightarrow{\text{Proposition 31.2.8}} \mathsf{is\text{-}conn}_k(f) \xrightarrow{\text{Proposition 31.2.9}} \mathsf{is\text{-}equiv}_k(f)$$

showing that the class of k-connected maps is contained in the class of k-equivalences, and that the class of $(k+1)$-equivalences is contained in the class of k-connected maps. Neither of these implications reverses.

Proposition 31.2.8. *Any k-connected map is a k-equivalence.*

Proposition 31.2.9. *Any $(k+1)$-equivalence is k-connected.*

Proof. Consider a $(k+1)$-equivalence $f : X \to Y$. Recall that the map $\|f\|_{k+1}$ comes equipped with a homotopy $H : \|f\|_{k+1} \circ \eta \sim \eta \circ f$ witnessing that the square

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "\eta"'] & Y \arrow[d, "\eta"] \\
\|X\|_{k+1} \arrow[r, "\|f\|_{k+1}"'] & \|Y\|_{k+1}
\end{tikzcd}$$

commutes. We be using this homotopy, and we will use Theorem 31.2.6 to show that f is k-connected. Thus, our goal is to show that

$$- \circ f : \left(\prod_{(y:Y)} P(y)\right) \to \left(\prod_{(x:X)} P(f(x))\right)$$

is an equivalence for any family P of k-types over Y.

Note that any family P of k-types over Y extends to a family $\tilde{P}$ of k-types over $\|Y\|_{k+1}$, since any univalent universe of k-types that contains P is itself a $(k+1)$-type by Exercise 12.1. The extended family $\tilde{P}$ of k-types over $\|Y\|_{k+1}$ comes equipped with a family of equivalences

$$e : \prod_{(y:Y)} \tilde{P}(\eta(y)) \simeq P(y).$$

Now consider the commuting diagram

$$\begin{tikzcd}
& \prod_{(y:\|Y\|_{k+1})} \tilde{P}(y) \arrow[r, "- \circ \|f\|_{k+1}"] \arrow[dddl, "- \circ \eta"'] & \prod_{(x:\|X\|_{k+1})} \tilde{P}(\|f\|_{k+1}(x)) \arrow[dr, "- \circ \eta"] & \\
& & & \prod_{(x:X)} \tilde{P}(\|f\|_{k+1}(\eta(x))) \arrow[dd, "h \mapsto \lambda x.\, \mathsf{tr}_{\tilde{P}}(H(x), h(x))"] \\
& & & \\
\prod_{(y:Y)} \tilde{P}(\eta(y)) \arrow[dr, "h \mapsto \lambda y.\, e_y(h(y))"'] & & & \prod_{(x:X)} \tilde{P}(\eta(f(x))) \arrow[dl, "h \mapsto \lambda x.\, e_{f(x)}(h(x))"] \\
& \prod_{(y:Y)} P(y) \arrow[r, "- \circ f"'] & \prod_{(x:X)} P(f(x)). &
\end{tikzcd}$$

This diagram commutes by the homotopy

$$\lambda h.\, \mathsf{eq\text{-}htpy}(\lambda x.\, \mathsf{ap}_{e(f(x))}(\mathsf{apd}_h(H(x)))^{-1}).$$

In this diagrams all the maps pointing downwards are equivalences for obvious reasons: the two maps $- \circ \eta$ are equivalences since $\tilde{P}$ is a family of k-types, and the remaining three maps pointing downwards are all postcomposing with an equivalence. The top map is an equivalence since $\|f\|_{k+1}$ is assumed to be an equivalence. Thus we conclude that the bottom map $- \circ f$ is an equivalence. □

The 3-for-2 property

An important distinction between the class of k-equivalences and the class of k-connected maps is that the k-equivalences satisfy the 3-for-2 property, while the k-connected maps do not.

Remark 31.2.10. It is not hard to see that the k-connected maps don't satisfy the 3-for-2 property. For example, consider the following commuting triangle

$$\begin{tikzcd} \mathbf{S}^1 \arrow[rr, "d_2"] \arrow[dr] & & \mathbf{S}^1 \arrow[dl] \\ & \mathbf{1}, & \end{tikzcd}$$

where $d_2 : \mathbf{S}^1 \to \mathbf{S}^1$ is the degree 2 map. Since the circle is a 0-connected type, it follows that the maps $\mathbf{S}^1 \to \mathbf{1}$ are 0-connected. However, the fiber of d_2 at the base point is equivalent to the booleans, which is a non-contractible set so it is certainly not 0-connected.

Lemma 31.2.11. *The k-equivalences satisfy the 3-for-2 property, i.e., for any commuting triangle*

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X, & \end{tikzcd}$$

if any two of the three maps are k-equivalences, then so is the third.

Proof. This follows immediately from the fact that equivalences satisfy the 3-for-2 property. □

Proposition 31.2.12. *Consider a commuting triangle*

$$\begin{tikzcd} A \arrow[rr, "h"] \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\ & X & \end{tikzcd}$$

with $H : f \sim g \circ h$. The following three statements hold:

(i) If f and h are k-connected, then g is k-connected.

(ii) If g and h are k-connected, then f is k-connected.

(iii) If f and g are k-connected, then h is a k-equivalence.

Proof. The first two statements combined assert that if h is k-connected, then f is k-connected if and only if g is k-connected. To see that this equivalence holds, consider for any family P of k-truncated types over X the commuting square

$$\begin{tikzcd} \prod_{(x:X)} P(x) \arrow[r, "-\circ g"] \arrow[d, "-\circ f"'] & \prod_{(b:B)} P(g(b)) \arrow[d, "-\circ h"] \\ \prod_{(a:A)} P(f(a)) \arrow[r, "\lambda s.\, \lambda a.\, \mathsf{tr}_P(H(a), s(a))"'] & \prod_{(a:A)} P(g(h(a))) \end{tikzcd}$$

In this square, the bottom map is given by postcomposing with the family of equivalences $\mathsf{tr}_P(H(a))$ indexed by $a : A$, so it is an equivalence. The map on the right is an equivalence by Theorem 31.2.6, using the assumption that h is a k-connected map. The square commutes by the homotopy

$$\lambda s.\, \mathsf{eq\text{-}htpy}\big(\lambda a.\, \mathsf{apd}_s(H(a))\big).$$

Therefore it follows that the precomposition map $- \circ f$ is an equivalence if and only if the precomposition map $- \circ g$ is. By Theorem 31.2.6 we conclude that f is connected if and only if g is. This proves statements (i) and (ii).

Statement (iii) follows from the facts that any k-connected map is a k-equivalence by **??** and that the k-equivalences satisfy the 3-for-2 property **??**. □

The action on homotopy groups

Theorem 31.2.13. *Consider a map $f : X \to Y$, and suppose that $k \geq -1$. The following are equivalent:*

(i) The map f is a k-equivalence.

(ii) The map f is a (-1)-equivalence, and for every $0 \leq i \leq k$ and every $x : X$, the induced group homomorphism

$$\pi_i(f, x) : \pi_i(X, x) \to \pi_i(Y, f(x))$$

is an isomorphism.

Definition 31.2.14. A map $f : X \to Y$ is said to be a **weak equivalence** if it is a 0-equivalence, and it induces an isomorphism

$$\pi_i(f, x) : \pi_i(X, x) \cong \pi_i(Y, f(x))$$

on homotopy groups, for every $x : X$ and every $i \geq 1$.

The following corollary is an instance of Whitehead's principle, which asserts that a map between any two spaces is a homotopy equivalence if and only if it is a weak equivalence. Thus, by the following corollary, Whitehead's principle holds for k-types.

Corollary 31.2.15. *Consider two k-types X and Y, and consider a map $f : X \to Y$ between them. Then the following are equivalent:*

(i) The map f is an equivalence.

(ii) The map f is a weak equivalence.

Theorem 31.2.16. *Consider a map $f : X \to Y$. The following are equivalent:*

(i) The map f is $(k+1)$-connected.

(ii) The map f is surjective, and for each $x, x' : X$ the action on paths

$$\mathsf{ap}_f : (x = x') \to (f(x) = f(x'))$$

is k-connected.

Theorem 31.2.17. *Consider a surjective map $f : X \to Y$. The following are equivalent:*

(i) The map f is k-connected.

(ii) The induced maps on loop spaces

$$\Omega(f, x) : \Omega(X, x) \to \Omega(Y, f(x))$$

is $(k-1)$-connected for every $x : X$.

(iii) The induced maps on homotopy groups

$$\pi_i(f,x) : \pi_i(X,x) \to \pi_i(Y,f(x))$$

are isomorphisms for $0 \leq i \leq k$*, and it is surjective for* $i = k+1$.

Remark 31.2.18. If $f : X \to Y$ is a pointed map between connected types, then conditions (ii) and (iii) in **??** only have to be checked at the base point.

31.3 Orthogonality

The idea of orthogonality is that a map $f : A \to B$ is left orthogonal to a map $g : X \to Y$ if for every commuting square of the form

$$\begin{tikzcd} A \arrow[r, "h"] \arrow[d, "f"'] & X \arrow[d, "g"] \\ B \arrow[r, "i"'] & Y, \end{tikzcd}$$

with $H : (i \circ f) \sim (g \circ h)$, the type of diagonal fillers is contractible. The type of diagonal fillers is the type of maps $j : B \to X$ equipped with homotopies

$$K : j \circ f \sim h$$

$$L : g \circ j \sim i$$

and a homotopy M witnessing that the triangle

$$\begin{tikzcd} g \circ j \circ f \arrow[rr, "g \cdot K"] \arrow[dr, "L \cdot f"'] & & h \circ g \\ & i \circ f \arrow[ur, "H"'] & \end{tikzcd}$$

commutes. A slicker way to express this condition is to assert that the map

$$(B \to X) \to \textstyle\sum_{(h:A\to X)}\sum_{(i:B\to Y)} i \circ f \sim g \circ h$$

given by $j \mapsto (j \circ f, g \circ j, \mathsf{refl\text{-}htpy})$ is an equivalence. Indeed, the type of triples (h, i, H) in the codomain is the type of commuting squares with respect to which we stated the orthogonality condition. Now we may even recognize the above map as a gap map of a commuting square, and we arrive at our actual definition of orthogonality.

Definition 31.3.1. A map $f : A \to B$ is said to be **left orthogonal** to a map $g : X \to Y$, or equivalently the map g is said to be **right orthogonal** to f, if the commuting square

$$\begin{tikzcd} X^B \arrow[r, "- \circ f"] \arrow[d, "g \circ -"'] & X^A \arrow[d, "g \circ -"] \\ Y^B \arrow[r, "- \circ f"'] & Y^A \end{tikzcd}$$

is a pullback square.

Theorem 31.3.2. *Let $f : A \to B$ be a map. The following are equivalent:*

(i) The map f is k-connected.

(ii) The map f is left orthogonal to every k-truncated map. is a pullback square.

Theorem 31.3.3. *Let $f : A \to B$ be a map. The following are equivalent:*

(i) The map f is a k-equivalence.

(ii) The map f is left orthogonal to every map between k-truncated types.

(iii) The map f is left orthogonal to every map $g : X \to Y$ for which the naturality square

$$\begin{tikzcd}
X \arrow[r, "g"] \arrow[d, "\eta"'] & Y \arrow[d, "\eta"] \\
\|X\|_k \arrow[r, "\|g\|_k"'] & \|Y\|_k
\end{tikzcd}$$

*is a pullback square. Such maps are called k-**étale**.*

31.4 The connectedness of suspensions

We will use connected maps to prove the connectedness of suspensions.

Proposition 31.4.1. *Consider a pushout square*

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\
A \arrow[r, "i"'] & X.
\end{tikzcd}$$

If the map $f : S \to A$ is k-connected, then so is the map $j : B \to X$.

Proof. We claim that the map $j : B \to X$ is left orthogonal to any k-truncated map $p : Y \to Z$, which is equivalent to the property that j is k-connected. To see that j is left orthogonal to p, consider the commuting cube

$$\begin{tikzcd}
& Y^X \arrow[dl] \arrow[d] \arrow[dr] & \\
Y^A \arrow[d] \arrow[dr] & Y^B \arrow[dl, crossing over] \arrow[dr] & Z^X \arrow[dl, crossing over] \arrow[d] \\
Y^S \arrow[dr] & Z^A \arrow[d] & Z^B \arrow[dl] \\
& Z^S. &
\end{tikzcd}$$

In this cube, the front left square is a pullback square because the map $f : S \to A$ is assumed to be k-connected, and therefore it is left orthogonal to the k-truncated map p. The back left and front right squares are pullback squares by the pullback property of pushouts. Therefore it follows that the back right square is a pullback square. This shows that j is left orthogonal to p. $\square$

Lemma 31.4.2. *A pointed type X is $(k+1)$-connected if and only if the point inclusion*

$$\mathbf{1} \to X$$

is a k-connected map.

Proof. Since X is assumed to have a base point $x_0 : X$, it follows that X is $(k+1)$-connected if and only if its identity types $(x = y)$ are k-connected. Now the claim follows from the fact that there is an equivalence

$$\mathsf{fib}_{\mathsf{const}_{x_0}}(y) \simeq (x_0 = y).$$ ☐

Theorem 31.4.3. *If X is an k-connected type, then its suspension ΣX is $(k+1)$-connected.*

Proof. The type X is k-connected if and only if the map $\mathsf{const}_\star : X \to \mathbf{1}$ is a k-connected map. Recall that the suspension of X is a pushout

$$\begin{tikzcd} X \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{const}_\star"'] & \mathbf{1} \arrow[d, "\mathsf{S}"] \\ \mathbf{1} \arrow[r, "\mathsf{N}"'] & \Sigma X. \end{tikzcd}$$

Therefore we see by Proposition 31.4.1 that the point inclusions $\mathsf{N}, \mathsf{S} : \mathbf{1} \to \Sigma X$ are both k-connected maps. By Lemma 31.4.2 it follows that ΣX is a $(k+1)$-connected type. ☐

Corollary 31.4.4. *The n-sphere is $(n-1)$-connected.*

Proof. The 0-sphere is (-1)-connected, since it contains a point. Thus the claim follows by induction on $n : \mathbb{N}$, using Theorem 31.4.3. ☐

31.5 The join connectivity theorem

Theorem 31.5.1. *If X is k-connected and Y is l-connected, then their join $X * Y$ is $(k+l+2)$-connected.*

Theorem 31.5.2. *Consider a pullback square*

$$\begin{tikzcd} C \arrow[r] \arrow[d] & B \arrow[d] \\ A \arrow[r] & X. \end{tikzcd}$$

If the maps $A \to X$ and $B \to X$ are k- and l-connected, respectively, then the map $A \sqcup^C B \to X$ is $(k+l+2)$-connected.

Theorem 31.5.3. *The connected maps contain the equivalences, are closed under coproducts, pushouts, retracts, and transfinite compositions.*

Exercises

31.1 Show that every type is equivalent to a disjoint union of connected components, i.e., show that for every type X there is a family of connected types B_i by a set I, with an equivalence

$$X \simeq \textstyle\sum_{(i:I)} B_i.$$

31.2 Let $f : A \to_* B$ be a pointed map between pointed n-connected types, for $n \geq -1$. Show that the following are equivalent:

(i) f is an equivalence.
(ii) $\Omega^{n+1}(f)$ is an equivalence.

31.3 Show that if

$$\begin{array}{ccc} A & \longrightarrow & B \\ {\scriptstyle f}\big\downarrow & & \big\downarrow{\scriptstyle g} \\ X & \longrightarrow & Y \end{array}$$

is k-**cocartesian** in the sense that the cogap map is k-connected, then the map $\mathsf{cofib}(f) \to \mathsf{cofib}(g)$ is k-connected.

31.4 Show that if $f : X \to Y$ is a k-connected map, then so is

$$\|f\|_l : \|X\|_l \to \|Y\|_l$$

for any $l \geq -2$.

31.5 Consider a commuting square

$$\begin{array}{ccc} A & \longrightarrow & B \\ {\scriptstyle f}\big\downarrow & & \big\downarrow{\scriptstyle g} \\ X & \longrightarrow & Y \end{array}$$

(a) Show that if the square is k-cartesian and g is k-connected, then so is f.
(b) Show that if f is k-connected and g is $(k+1)$-connected, then the square is k-cartesian.

31.6 (a) Show that any sequential colimit of k-connected types is again k-connected.
(b) Show that if every map in a type sequence

$$A_0 \longrightarrow A_1 \longrightarrow A_2 \longrightarrow \cdots$$

is k-connected, then so is the transfinite composition $A_0 \to A_\infty$.

31.7 Recall that a commuting square is called k-cartesian, if its gap map is k-connected. Show that $(k+1)$-truncation preserves l-cartesian squares for any $l \leq k$, i.e., show that for any $l \leq k$, if a square

$$\begin{array}{ccc} C & \xrightarrow{q} & B \\ {\scriptstyle p}\big\downarrow & & \big\downarrow{\scriptstyle g} \\ A & \xrightarrow[f]{} & X. \end{array}$$

is l-cartesian, then the square

$$\begin{array}{ccc} \|C\|_{k+1} & \xrightarrow{\|q\|_{k+1}} & \|B\|_{k+1} \\ {\scriptstyle \|p\|_{k+1}}\big\downarrow & & \big\downarrow{\scriptstyle \|g\|_{k+1}} \\ \|A\|_{k+1} & \xrightarrow[\|f\|_{k+1}]{} & \|X\|_{k+1} \end{array}$$

is l-cartesian.

31.8 Generalize Remark 31.2.10 to show that for every $k \geq -1$, the k-connected maps do not satisfy the 3-for-2 property.

31.9 Consider a commuting square

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & B \arrow[d, "g"] \\ X \arrow[r] & Y \end{tikzcd}$$

Show that the following are equivalent:

(i) The map $A \to X \times_Y B$ is n-connected. In this case the square is called **n-cartesian**.

(ii) For each $x : X$ the map

$$\mathsf{fib}_f(x) \to \mathsf{fib}_g(f(x))$$

is n-connected.

31.10 Consider a map $f : A \to B$. Show that the following are equivalent:

(i) The map f is a weak equivalence.

(ii) The map f is ∞-connected, in the sense that f is k-connected for each k.

(iii) The map f is left orthogonal to any map between truncated types of any truncation level.

(iv) The map f is left orthogonal to any truncated map, for any truncation level.

Thus we see that, while the classes of k-connected maps and k-equivalences differ for finite $k \geq -1$, they come to agree at ∞.

31.11 Consider a pointed $(k+1)$-connected type X. Show that every k-truncated map $f : A \to X$ trivializes, in the sense that there is a k-type B and an equivalence $e : A \simeq X \times B$ for which the triangle

$$\begin{tikzcd} A \arrow[rr, "e"] \arrow[dr, "f"'] & & X \times B \arrow[dl, "\mathsf{pr}_1"] \\ & X & \end{tikzcd}$$

commutes.

31.12 Consider a k-equivalence $f : B' \to B$. Show that the base-change functor induces an equivalence

$$\Big(\textstyle\sum_{(E:\mathcal{U})}\sum_{(p:E\to B)}\mathsf{is\text{-}etale}_k(p)\Big) \simeq \Big(\textstyle\sum_{(E':\mathcal{U})}\sum_{(p':E'\to B')}\mathsf{is\text{-}etale}_k(p')\Big).$$

In other words, for every k-étale map $p' : E' \to B'$ there is a unique k-étale map $p : E \to B$ equipped with a map $q : E' \to E$ such that the square

$$\begin{tikzcd} E' \arrow[r, dotted, "q"] \arrow[d, "p'"'] & E \arrow[d, dotted, "p"] \\ B' \arrow[r, "f"'] & B \end{tikzcd}$$

commutes and is a pullback square. In this sense k-étale maps descend along k-equivalences.

32 The Blakers-Massey theorem

The Blakers-Massey theorem is a connectivity theorem which can be used to prove the Freudenthal suspension theorem, giving rise to the field of *stable homotopy theory*. It was proven in the setting of homotopy type theory by Lumsdaine et al, and their proof was the first that was given entirely in an elementary way, using only constructions that are invariant under homotopy equivalence.

32.1 The Blakers-Massey theorem

Consider a span $A \leftarrow S \rightarrow B$, consisting of an m-connected map $f : S \rightarrow A$ and an n-connected map $g : S \rightarrow B$. We take the pushout of this span, and subsequently the pullback of the resulting cospan, as indicated in the diagram

$$\begin{tikzcd}
S \arrow[dr, dotted, "u"] \arrow[drr, bend left, "g"] \arrow[ddr, bend right, "f"'] & & \\
& A \times_{(A \sqcup^S B)} B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"] & B \arrow[d, "\mathsf{inr}"] \\
& A \arrow[r, "\mathsf{inl}"'] & A \sqcup^S B.
\end{tikzcd} \tag{32.1}$$

The universal property of the pullback determines a unique map $u : S \rightarrow A \times_{(A \sqcup^S B)} B$ as indicated.

Theorem 32.1.1 (Blakers-Massey)**.** *The map $u : S \rightarrow A \times_{(A \sqcup^S B)} B$ of Eq. (32.1) is $(n+m)$-connected.*

32.2 The Freudenthal suspension theorem

Theorem 32.2.1. *If X is a k-connected pointed type, then the canonical map*

$$X \rightarrow \Omega(\Sigma X)$$

is $2k$-connected.

Theorem 32.2.2. $\pi_n(\mathbf{S}^n) = \mathbb{Z}$ *for* $n \geq 1$.

32.3 Higher groups

Recall that types in HoTT may be viewed as ∞-groupoids: elements are objects, paths are morphisms, higher paths are higher morphisms, etc.

It follows that *pointed connected* types B may be viewed as higher groups, with **carrier** ΩB. The neutral element is the identity path, the group operation is given by path composition, and higher paths witness the unit and associativity laws. Of course, these higher paths are themselves subject to further laws, etc., but the beauty of the type-theoretic definition is that we don't have to worry about that: all the (higher) laws follow from the rules of the identity types. Writing G for the carrier ΩB, it is common to write BG for the pointed connected type B, which comes equipped with an identification $G = \Omega BG$. We call BG the **delooping** of G.

The type of pointed types is $\mathcal{U}_{\mathsf{pt}} :\equiv \sum_{(A:\mathcal{U})} A$. The type of n-truncated types is $\mathcal{U}^{\leq n} :\equiv \sum_{(A:\mathcal{U})} \mathsf{is\text{-}trunc}_n A$ and for n-connected types it is $\mathcal{U}^{>n} :\equiv \sum_{(A:\mathcal{U})} \mathsf{is\text{-}conn}_n(A)$. We will combine these notations as needed.

Definition 32.3.1. We define the type of **higher groups**, or **∞-groups**, to be

$$\infty\mathsf{Grp} :\equiv \textstyle\sum_{(G:\mathcal{U})}\sum_{(BG:\mathcal{U}_{\mathsf{pt}}^{>0})} G \simeq \Omega BG.$$

When G is an ∞-group, we also write G for its first projection, called the **carrier** of G.

Remark 32.3.2. Note that we have equivalences

$$\begin{aligned}\infty\mathsf{Grp} &\equiv \textstyle\sum_{(G:\mathcal{U})}\sum_{(BG:\mathcal{U}_{\mathsf{pt}}^{>0})} G \simeq \Omega BG \\ &\simeq \textstyle\sum_{(G:\mathcal{U}_{\mathsf{pt}})}\sum_{(BG:\mathcal{U}_{\mathsf{pt}}^{>0})} G \simeq_{\mathsf{pt}} \Omega BG \\ &\simeq \mathcal{U}_{\mathsf{pt}}^{>0}\end{aligned}$$

for the type of higher groups.

Automorphism groups form a major class of examples of ∞-groups. Given *any* type A and any object $a : A$, the automorphism group at a is defined as **automorphism group** $\mathrm{Aut}\, a :\equiv (a = a)$. This is indeed an ∞-group, because it is the loop space of the connected component of A at a, i.e. we define $\mathsf{BAut}a :\equiv \mathrm{im}(a : 1 \to A) = (x : A) \times \|a = x\|_{-1}$. From this definition it is immediate that $\mathrm{Aut}\, a = \Omega\mathsf{BAut}a$, so we see that $\mathrm{Aut}\, a$ is indeed an example of an ∞-group.

If we take $A = \mathsf{Set}$, we get the usual symmetric groups $S_n :\equiv \mathsf{Aut}(\mathsf{Fin}(n))$, where $\mathsf{Fin}(n)$ is a set with n elements. (Note that $BS_n = \mathsf{BAut}(\mathsf{Fin}(n))$ is the type of all n-element sets.)

We recover the ordinary set-level groups by requiring that G is a 0-type, or equivalently, that BG is a 1-type. This leads us to introduce:

Definition 32.3.3. We define the type of **groupal** $(n-1)$**-groupoids**, or n**-groups**, to be

$$n\mathsf{Grp} :\equiv \textstyle\sum_{(G:\mathcal{U}_{\mathsf{pt}}^{<n})}\sum_{(BG:\mathcal{U}_{\mathsf{pt}}^{>0})} G \simeq_{\mathsf{pt}} \Omega BG.$$

We write Grp for the type of 1-groups.

The type of n-groups is therefore equivalent to the type of pointed connected $(n+1)$-types. Note that if A is an $(n+1)$-type, then $\mathrm{Aut}\, a$ is an $(n+1)$-group because $\mathrm{Aut}\, a$ is n-truncated.

For example, the integers $\mathbb{Z}$ as an additive group are from this perspective represented by their delooping $\mathrm{B}\mathbb{Z} = \mathrm{S}^1$, i.e., the circle. Indeed, any set-level group G is represented as its delooping $BG :\equiv K(G, 1)$.

Moving across the homotopy hypothesis, for every pointed type (X, x) we have the **fundamental ∞-group of** X, $\Pi_\infty(X, x) :\equiv \mathrm{Aut}\, x$. Its $(n-1)$-truncation (an instance of decategorification, see §32.4) is the **fundamental n-group of** X, $\Pi_n(X, x)$, with corresponding delooping $\mathrm{B}\Pi_n(X, x) = \|\mathsf{BAut}x\|_n$.

Double loop spaces are more well-behaved than mere loop spaces. For example, they are commutative up to homotopy by the Eckmann-Hilton argument [3, Theorem 2.1.6]. Triple loop spaces are even better behaved than double loop spaces, and so on.

Definition 32.3.4. A type G is said to be k**-tuply groupal** if it comes equipped with a k**-fold delooping**, i.e. a pointed k-connected $B^kG : \mathcal{U}_{\mathsf{pt}}^{\geq k}$ and an equivalence $G \simeq \Omega^k B^k G$.

Mixing the two directions, we also define

$$\begin{aligned}(n,k)\mathrm{GType} &:\equiv \textstyle\sum_{(G:\mathcal{U}_{\mathsf{pt}}^{\leq n})}\sum_{(B^kG:\mathcal{U}_{\mathsf{pt}}^{\geq k})} G \simeq_{\mathsf{pt}} \Omega^k B^k G \\ &\simeq \mathcal{U}_{\mathsf{pt}}^{\geq k, \leq n+k}\end{aligned}$$

Table V.1: Periodic table of k-tuply groupal n-groupoids.

$k \setminus n$	0	1	2	$\cdots$	∞
0	pointed set	pointed groupoid	pointed 2-groupoid	$\cdots$	pointed ∞-groupoid
1	group	2-group	3-group	$\cdots$	∞-group
2	abelian group	braided 2-group	braided 3-group	$\cdots$	braided ∞-group
3	— ″ —	symmetric 2-group	sylleptic 3-group	$\cdots$	sylleptic ∞-group
4	— ″ —	— ″ —	symmetric 3-group	$\cdots$	?? ∞-group
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Ω	— ″ —	— ″ —	— ″ —	$\cdots$	connective spectrum

for the type of **k-tuply groupal n-groupoids**[1]. We allow taking $n = \infty$, in which case the truncation requirement is simply dropped.

Note that $n\mathsf{Grp} = (n-1,1)\mathrm{GType}$. This shift in indexing is slightly annoying, but we keep it to stay consistent with the literature.

Note that for each $k \geq 0$ there is a forgetful map

$$(n, k+1)\mathrm{GType} \to (n,k)\mathrm{GType},$$

given by $B^{k+1}G \mapsto \Omega B^{k+1}G$, defining a sequence

$$\cdots \longrightarrow (n,2)\mathrm{GType} \longrightarrow (n,1)\mathrm{GType} \longrightarrow (n,0)\mathrm{GType}.$$

Thus we define $(n,\infty)\mathrm{GType}$ as the limit of this sequence:

$$\begin{aligned}(n,\infty)\mathrm{GType} &:\equiv \mathsf{lim}_k (n,k)\mathrm{GType} \\ &\simeq \textstyle\sum_{(B^- G:\prod_{(k:\mathbb{N})}\mathcal{U}_{\mathsf{pt}}^{\geq k,\leq n+k})}\prod_{(k:\mathbb{N})} B^k G \simeq_{\mathsf{pt}} \Omega B^{k+1} G.\end{aligned}$$

In §32.4 we prove the stabilization theorem (Theorem 32.4.10), from which it follows that $(n,\infty)\mathrm{GType} = (n,k)\mathrm{GType}$ for $k \geq n+2$.

The type $(\infty,\infty)\mathrm{GType}$ is the type of **stably groupal ∞-groups**, also known as **connective spectra**. If we also relax the connectivity requirement, we get the type of all spectra, and we can think of a spectrum as a kind of ∞-groupoid with k-morphisms for all $k \in \mathbb{Z}$.

The double hierarchy of higher groups is summarized in Table V.1. We shall prove the correctness of the $n = 0$ column in **??**.

A homomorphism between higher groups is any function that can be suitably delooped.

Definition 32.3.5. For $G, H : (n,k)\mathrm{GType}$, we define

$$\begin{aligned}\mathsf{hom}_{(n,k)}(G,H) &:\equiv \textstyle\sum_{(h:G\to_{\mathsf{pt}}H)}\sum_{(B^k h:B^k G\to_{\mathsf{pt}}B^k H)}\Omega^k(B^k h) \sim_{\mathsf{pt}} h \\ &\simeq (B^k h : B^k G \to_{\mathsf{pt}} B^k H).\end{aligned}$$

For (connective) spectra we need pointed maps between all the deloopings and pointed homotopies showing they cohere.

[1] This is called $n\mathcal{U}_k$ in **[BaezDolan1998]**, but here we give equal billing to n and k, and we add the "G" to indicate group-structure.

Note that if $h, k : G \to H$ are homomorphisms between set-level groups, then h and k are **conjugate** if $Bh, Bk : BG \to_{\mathsf{pt}} BH$ are **freely** homotopic (i.e., equal as maps $BG \to BH$).

Also observe that

$$\begin{aligned}\pi_j(B^k G \to_{\mathsf{pt}} B^k H) &\simeq \|B^k G \to_{\mathsf{pt}} \Omega^j B^k H\|_0 \\ &\simeq \|\Sigma^j B^k G \to_{\mathsf{pt}} B^k H\|_0 \\ &\simeq 0\end{aligned}$$

for $j > n$, which suggests that $\mathsf{hom}_{(n,k)}(G, H)$ is n-truncated. To prove this, we deviate slightly from the approach in **[BuchholtzDoornRijke]** and use the following intermediate result.

32.4 The stabilization theorem for higher groups

Definition 32.4.1. The **decategorification** $\mathrm{Decat}\, G$ of a k-tuply groupal $(n+1)$-group is defined to be the k-tuply groupal n-group $\|G\|_{n-1}$, which has delooping $\|B^k G\|_{n+k-1}$. Thus, decategorification is an operation

$$\mathrm{Decat} : (n, k)\mathrm{GType} \to (n-1, k)\mathrm{GType}.$$

The functorial action of Decat is defined in the expected way. We also define the **∞-decategorification** ∞-Decat$G$ of a k-tuply groupal ∞-group as the k-tuply groupal n-group $\|G\|_n$, which has delooping $\|B^k G\|_{n+k}$.

Definition 32.4.2. The **discrete categorification** $\mathrm{Disc}\, G$ of a k-tuply-groupal $(n+1)$-group is defined to be the same ∞-group G, now considered as a k-tuply groupal $(n+2)$-group. Thus, the discrete categorification is an operation

$$\mathrm{Disc} : (n, k)\mathrm{GType} \to (n+1, k)\mathrm{GType}.$$

Similarly, the **discrete ∞-decategorification** ∞-Disc$G$ of a k-tuply groupal $(n+1)$-group is defined to be the same group, now considered as a k-tuply groupal ∞-group.

Remark 32.4.3. The decategorification and discrete categorification functors make the $(n+1)$-category (n, k)GType a reflective sub-$(\infty, 1)$-category of $(n+1, k)$GType. That is, there is an adjunction Decat $\dashv$ Disc. These properties are straightforward consequences of the universal property of truncation. Similarly, we have ∞-Decat $\dashv$ ∞-Disc such that the counit induces an isomorphism ∞-Decat $\circ$ ∞-Disc $=$ id.

For the next constructions, we need the following properties.

Definition 32.4.4. For $A : \mathcal{U}_{\mathsf{pt}}$ we define the **n-connected cover** of A to be $A\langle n\rangle :\equiv \mathsf{fib}_{A \to \|A\|_n}$. We have the projection $p_1 : A\langle n\rangle \to_{\mathsf{pt}} A$.

Lemma 32.4.5. *The universal property of the n-connected cover states the following. For any n-connected pointed type B, the pointed map*

$$(B \to_{\mathsf{pt}} A\langle n\rangle) \to_{\mathsf{pt}} (B \to_{\mathsf{pt}} A),$$

given by postcomposition with p_1, is an equivalence.

Proof. Given a map $f : B \to_{\mathsf{pt}} A$, we can form a map $\widetilde{f} : B \to A\langle n\rangle$. First note that for $b : B$ the type $|fb|_n =_{\|A\|_n} |\mathsf{pt}|_n$ is $(n-1)$-truncated and inhabited for $b = \mathsf{pt}$. Since B is n-connected, the universal property for connected types shows that we can construct a $qb : |fb|_n = |\mathsf{pt}|_n$ for all b such that $q_0 : qb_0 \cdot \mathsf{ap}_{|-|_n}(f_0) = 1$. Then we can define the map $\widetilde{f}(b) :\equiv (fb, qb)$. Now $\widetilde{f}$ is pointed, because $(f_0, q_0) : (fb_0, qb_0) = (a_0, 1)$.

Now we show that this is indeed an inverse to the given map. On the one hand, we need to show that if $f : B \to_{\mathsf{pt}} A$, then $\mathsf{pr}_1 \circ \widetilde{f} = f$. The underlying functions are equal because they both send b to $f(b)$. They respect points in the same way, because $\mathsf{ap}p_1(\widetilde{f_0}) = f_0$. The proof that the other composite is the identity follows from a computation using fibers and connectivity, which we omit here, but can be found in the formalization. □

The next reflective sub-$(\infty, 1)$-category is formed by looping and delooping.

looping $\Omega : (n,k)\text{GType} \to (n-1, k+1)\text{GType}$
$\langle G, B^k G\rangle \mapsto \langle \Omega G, B^k G\langle k\rangle\rangle$

delooping $\mathrm{B} : (n,k)\text{GType} \to (n+1, k-1)\text{GType}$
$\langle G, B^k G\rangle \mapsto \langle \Omega^{k-1} B^k G, B^k G\rangle$

We have $\mathrm{B} \dashv \Omega$, which follows from Lemma 32.4.5 and $\Omega \circ \mathrm{B} = \mathsf{id}$, which follows from the fact that $A\langle n\rangle = A$ if A is n-connected.

The last adjoint pair of functors is given by stabilization and forgetting. This does not form a reflective sub-$(\infty, 1)$-category.

forgetting $F : (n,k)\text{GType} \to (n, k-1)\text{GType}$
$\langle G, B^k G\rangle \mapsto \langle G, \Omega B^k G\rangle$

stabilization $S : (n,k)\text{GType} \to (n, k+1)\text{GType}$
$\langle G, B^k G\rangle \mapsto \langle SG, \|\Sigma B^k G\|_{n+k+1}\rangle$,
where $SG = \|\Omega^{k+1}\Sigma B^k G\|_n$

We have the adjunction $S \dashv F$ which follows from the suspension-loop adjunction $\Sigma \dashv \Omega$ on pointed types.

The next main goal in this section is the stabilization theorem, stating that the ditto marks in Table V.1 are justified.

The following corollary is almost [3, Lemma 8.6.2], but proving this in Book HoTT is a bit tricky. See the formalization for details.

Lemma 32.4.6 (Wedge connectivity)**.** *If $A : \mathcal{U}_{\mathsf{pt}}$ is n-connected and $B : \mathcal{U}_{\mathsf{pt}}$ is m-connected, then the map $A \vee B \to A \times B$ is $(n+m)$-connected.*

Let us mention that there is an alternative way to prove the wedge connectivity lemma: Recall that if A is n-connected and B is m-connected, then $A * B$ is $(n+m+2)$-connected [**joinconstruction**]. Hence the wedge connectivity lemma is also a direct consequence of the following lemma.

Lemma 32.4.7. *Let A and B be pointed types. The fiber of the wedge inclusion $A \vee B \to A \times B$ is equivalent to $\Omega A * \Omega B$.*

Proof. Note that the fiber of $A \to A \times B$ is ΩB, the fiber of $B \to A \times B$ is ΩA, and of course the fiber of $1 \to A \times B$ is $\Omega A \times \Omega B$. We get a commuting cube

$$\begin{tikzcd}
 & \Omega A \times \Omega B \arrow[dl] \arrow[d] \arrow[dr] & \\
\Omega B \arrow[d] \arrow[dr] & 1 \arrow[dl] \arrow[dr] & \Omega A \arrow[dl, crossing over] \arrow[d] \\
A \arrow[dr] & 1 \arrow[d] & B \arrow[dl] \\
 & A \times B &
\end{tikzcd}$$

in which the vertical squares are pullback squares.

By the descent theorem for pushouts it now follows that $\Omega A * \Omega B$ is the fiber of the wedge inclusion. ☐

The second main tool we need for the stabilization theorem is:

Theorem 32.4.8 (Freudenthal)**.** *If $A : \mathcal{U}_{\mathsf{pt}}^{\geq n}$ with $n \geq 0$, then the map $A \to \Omega\Sigma A$ is $2n$-connected.*

This is [3, Theorem 8.6.4].

The final building block we need is:

Lemma 32.4.9. *There is a pullback square*

$$\begin{tikzcd}
\Sigma\Omega A \arrow[r] \arrow[d, "\varepsilon_A"'] & A \vee A \arrow[d] \\
A \arrow[r, "\Delta"'] & A \times A
\end{tikzcd}$$

for any $A : \mathcal{U}_{\mathsf{pt}}$.

Proof. Note that the pullback of $\Delta : A \to A \times A$ along either inclusion $A \to A \times A$ is contractible. So we have a cube

$$\begin{tikzcd}
 & \Omega A \arrow[dl] \arrow[d] \arrow[dr] & \\
1 \arrow[d] \arrow[dr] & 1 \arrow[dl] \arrow[dr] & 1 \arrow[dl, crossing over] \arrow[d] \\
A \arrow[dr] & A \arrow[d, "\Delta"] & A \arrow[dl] \\
 & A \times A &
\end{tikzcd}$$

in which the vertical squares are all pullback squares. Therefore, if we pull back along the wedge inclusion, we obtain by the descent theorem for pushouts that the square in the statement is indeed a pullback square. ☐

Theorem 32.4.10 (Stabilization)**.** *If $k \geq n+2$, then $S : (n,k)\mathrm{GType} \to (n,k+1)\mathrm{GType}$ is an equivalence, and any $G : (n,k)\mathrm{GType}$ is an infinite loop space.*

Proof. We show that $F \circ S = \mathsf{id} = S \circ F : (n,k)\text{GType} \to (n,k)\text{GType}$ whenever $k \geq n+2$.

For the first, the unit map of the adjunction factors as

$$B^k G \to \Omega\Sigma B^k G \to \Omega\|\Sigma B^k G\|_{n+k+1}$$

where the first map is $2k-2$-connected by Freudenthal, and the second map is $n+k$-connected. Since the domain is $n+k$-truncated, the composite is an equivalence whenever $2k-2 \geq n+k$.

For the second, the counit map of the adjunction factors as

$$\|\Sigma\Omega B^k G\|_{n+k} \to \|B^k G\|_{n+k} \to B^k G,$$

where the second map is an equivalence. By the two lemmas above, the first map is $2k-2$-connected. □

For example, for $G : (0,2)\text{GType}$ an abelian group, we have $B^n G = K(G,n)$, an Eilenberg-MacLane space.

The adjunction $S \dashv F$ implies that the free group on a pointed set X is $\Omega\|\Sigma X\|_1 = \pi_1(\Sigma X)$. If X has decidable equality, ΣX is already 1-truncated. It is an open problem whether this is true in general.

Also, the abelianization of a set-level group $G : 1\mathsf{Grp}$ is $\pi_2(\Sigma BG)$. If $G : (n,k)\text{GType}$ is in the stable range ($k \geq n+2$), then $SFG = G$.

32.5 Eilenberg-Mac Lane spaces

Exercises

32.1 Show that if X is m-connected and $f : X \to Y$ is n-connected, then the map

$$X \to \mathsf{fib}_{m_f}(*)$$

where $m_f : Y \to M_f$ is the inclusion of Y into the cofiber of f, is $(m+n)$-connected.

32.2 Suppose that X is a connected type, and let $f : X \to Y$ be a map. Show that the following are equivalent:

(i) f is n-connected.

(ii) The mapping cone of f is $(n+1)$-connected.

32.3 Apply the Blakers-Massey theorem to the defining pushout square of the smash product to show that if A and B are m- and n-connected respectively, then there is a $(m+n+\min(m,n)+2)$-connected map

$$\Omega(A) * \Omega(B) \to \Omega(A \wedge B).$$

32.4 Show that the square

$$\begin{array}{ccc} \mathbf{1} & \longrightarrow & \mathbf{2} \\ \downarrow & & \downarrow \\ X & \longrightarrow & X+\mathbf{1} \end{array}$$

is both a pullback and a pushout. Conclude that the result of the Blakers-Massey theorem is not always sharp.

32.5 Show that for every pointed type X, and any $n : \mathbb{N}$, there is a fiber sequence

$$K(\pi_{n+1}(X), n+1) \hookrightarrow \|X\|_{n+1} \twoheadrightarrow \|X\|_n.$$

33 Higher group theory

33.1 The category of pointed connected 1-types

Proposition 33.1.1. *Consider a k-connected map $f : X \to Y$, and a family P of $(k+n)$-truncated types over Y, where $n \geq 0$. Then the precomposition map*

$$- \circ f : \Big(\prod_{(y:Y)} P(y)\Big) \to \Big(\prod_{(x:X)} P(f(x))\Big)$$

is $(n-2)$-truncated.

Proposition 33.1.2. *Consider a pointed $(k+1)$-connected type X, and a family $Y : X \to \mathcal{U}^{\leq n+k}$ of $(n+k)$-truncated types over X. Then the map*

$$\mathsf{ev\text{-}pt} : \Big(\prod_{(x:X)} Y(x)\Big) \to Y(\mathsf{pt})$$

induced by the point inclusion $\mathbf{1} \to X$, is an $(n-2)$-truncated map.

Proof. Note that we have a commuting triangle

$$\begin{tikzcd} & \Big(\prod_{(x:X)} Y(x)\Big) \arrow[dl, "-\circ \mathsf{const}_{\mathsf{pt}}"'] \arrow[dr, "\mathsf{ev\text{-}pt}"] & \\ \Big(\prod_{(t:\mathbf{1})} Y(\mathsf{pt})\Big) \arrow[rr, "\simeq", "\mathsf{ev\text{-}pt}"'] & & Y(\mathsf{pt}), \end{tikzcd}$$

so the map on the left is an $(n-2)$-truncated map if and only if the map on the right is. For the map on the left, the claim follows immediately from Proposition 33.1.1, since the point inclusion $\mathsf{const}_{\mathsf{pt}} : \mathbf{1} \to X$ is a k-connected map by **??**. □

Definition 33.1.3. If $X : \mathcal{U}_{\mathsf{pt}}$ and $Y : X \to \mathcal{U}_{\mathsf{pt}}$, then we introduce the type of **pointed sections**,

$$\prod^*_{(x:X)} Y(x) :\equiv \sum_{(s:\prod_{(x:X)} Y(x))} s(\mathsf{pt}) = \mathsf{pt}$$

This type is itself pointed by the trivial section $\lambda x.\, \mathsf{pt}$.

Corollary 33.1.4. *Consider a pointed k-connected type X, and a family $Y : X \to \mathcal{U}_{\mathsf{pt}}^{\leq n+k}$ of pointed $(n+k)$-truncated types over X. Then the type $\prod^*_{(x:X)} Y(x)$ is $(n-1)$-truncated.*

Proof. Note that we have a pullback square

$$\begin{tikzcd} \prod^*_{(x:X)} Y(x) \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\ \prod_{(x:X)} Y(x) \arrow[r, "\mathsf{ev\text{-}pt}"'] & Y(*), \end{tikzcd}$$

so the claim follows from the fact that $\mathsf{ev\text{-}pt}$ is an $(n-1)$-truncated map. □

Theorem 33.1.5. *The type $\mathsf{hom}_{(n,k)}(G, H)$ is an n-type for any $G, H : (n,k)\mathrm{GType}$.*

Proof. If X is $(k-1)$-connected, and Y is $(n+k)$-truncated, then the type of pointed maps $X \to_{\mathsf{pt}} Y$ is n-truncated. □

Corollary 33.1.6. *The type* $(n,k)\mathrm{GType}$ *is* $(n+1)$*-truncated.*

Proof. This follows immediately from the preceding corollary, as the type of equivalences $G \simeq H$ is a subtype of the homomorphisms from G to H. □

If $k \geq n+2$ (so we're in the stable range), then $\mathsf{hom}_{(n,k)}(G,H)$ becomes a stably groupal n-groupoid. This generalizes the fact that the homomorphisms between abelian groups form an abelian group.

Corollary 33.1.7. *The automorphism group* $\mathrm{Aut}\, G$ *of a higher group* $G : (n,k)\mathrm{GType}$ *is a* 1*-groupal* $(n+1)$*-group, equivalent to the automorphism group of the pointed type* $B^k G$*.*

Proposition 33.1.8. *For any two pointed n-connected* $(n+k+1)$*-truncated types X and Y, the type of pointed maps*

$$X \to_* Y$$

is k-truncated.

Corollary 33.1.9. *For any two pointed n-connected* $(n+1)$*-truncated types X and Y, the type of pointed maps*

$$X \to_* Y$$

is a set.

Theorem 33.1.10. *The pre-category of n-connected* $(n+1)$*-truncated types in a universe* $\mathcal{U}$ *is Rezk complete.*

33.2 Equivalences of categories

Definition 33.2.1. A **functor** is...

Definition 33.2.2. A functor $F : \mathcal{C} \to \mathcal{D}$ is an equivalence if ...

33.3 The equivalence of groups and pointed connected 1-types

Theorem 33.3.1. *The loop space functor*

$$\mathsf{Type}_0^1 \to \mathsf{Group}$$

is an equivalence of categories.

Overview of the axioms in this book

(i) We assumed the function extensionality axiom in Axiom 11.1.1. Function extensionality was later derived from the univalence axiom in Theorem 12.2.2.

(ii) We assumed the univalence axiom in Axiom 12.1.1.

(iii) We assumed that universes are closed under propositional truncations in Axiom 13.1.8. It was shown in **??** that any universe that is closed under homotopy pushouts is also closed under propositional truncations.

(iv) We assumed the type theoretic replacement axiom in Axiom 14.3.5. It was shown in Theorem 27.6.11 that replacement holds for any universe that is closed under homotopy pushouts.

(v) We assumed in **??** that the circle is a type in the base universe $\mathcal{U}_0$. In **??** the circle was shown to be a type in the base universe using the replacement axiom. It was shown that the In **??**, the circle was constructed as a pushout.

(vi) In **??** that universes are closed under pushouts.

Bibliography

[1] J. F. Adams. "On the nonexistence of elements of Hopf invariant one". In: *Bull. Amer. Math. Soc.* 64.5 (Sept. 1958), pp. 279–282. URL: `https://projecteuclid.org:443/euclid.bams/1183522680`.

[2] Erret Bishop. *Foundations of constructive analysis*. New York: McGraw-Hill Book Co., 1967, pp. xiii+370.

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

Index