Introduction to homotopy type theory

Egbert Rijke

Carnegie Mellon University
Pittsburgh PA
Spring 2018
Version: May 18, 2018

Contents

Syllabus

Essential course information

Course title	Introduction to Homotopy Type Theory
Instructor	Egbert Rijke Department of Philosophy Carnegie Mellon University
Course number	80-518, 80-818
Semester	Spring 2018
Website	`http://www.andrew.cmu.edu/user/erijke/hott/`
Lecture room	Baker Hall 150
Meeting time	Tue/Thu 12:00 - 1:20
Email	erijke@andrew.cmu.edu
Instructor's office	Baker Hall 148
Office Hours	Mon/Wed 5:00 - 6:00, or by appointment

Course description

Homotopy Type Theory (HoTT) is an emerging field of mathematics and computer science that extends Martin-Löf's dependent type theory by the addition of the univalence axiom and higher inductive types. In HoTT we think of types as spaces, dependent types as fibrations, and of the identity types as path spaces. We start the course by introducing type theory as a deductive system, and once the basic ingredients of homotopy type theory are in place we will mainly focus on *synthetic homotopy theory*, i.e. the development of homotopy theory in type theory.

Course material

We will roughly follow the book *Homotopy Type Theory: Univalent foundation of mathematics* [2], of which a PDF is freely available.

Some of the later results of synthetic homotopy theory can only be found in recent research papers. We will also use the PhD thesis of Guillaume Brunerie [1] as a resource.

Organization

Each session will consist of two parts: a 50 minute lecture and 30 minutes in which students present solutions to exercises provided with the previous lecture. These presentations are intended to be short (roughly 5 minutes) and focused to the problem at hand. Problem sets will be posted below with the lecture synopses.

Students are expected to:

(i) Present a solution when they are asked to do so (usually a week in advance). Graduate students will be asked to present more often than undergraduate students.

(ii) Per lecture, either correct a somewhat substantial mistake made by the instructor, or hand in a written solution for one exercise of their choice. Written solutions are to be handed in at the start of the next lecture for an A, or at the start of the next lecture after that for a B. Collaborations are encouraged, but solutions must be handed in individually. Presenting students hand in a written solution for the exercise they are asked to present.

Hints for the exercises will be presented by the instructor during office hours, a day before they have to be handed in.

Lecture 1

Dependent type theory

1.1 The primitive judgments of type theory

The theory of type dependency is formulated as a deductive system in which derivations establish that a given construction is well-formed. In any dependent type theory there are four **primitive judgments**:

(i) '*A is a well-formed* ***type*** *in context* Γ.'

(ii) '*A and B are* ***judgmentally equal types*** *in context* Γ.'

(iii) '*a is a well-formed* ***term*** *of type A in context* Γ.'

(iv) '*a and b are* ***judgmentally equal terms*** *of type A in context* Γ.'

The symbolic expressions for these four primitive judgments are as follows:

$$\begin{array}{ll} \Gamma \vdash A \text{ type} & \Gamma \vdash A \equiv B \text{ type} \\ \Gamma \vdash a : A & \Gamma \vdash a \equiv b : A. \end{array}$$

A **context** is an expression of the form

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

which we often simply write as $x_1 : A_1,\ x_2 : A_2,\ \ldots,\ x_n : A_n$, satisfying the condition that for each $1 \leq k \leq n$ we have that A_k is a well-formed type in context $x_1 : A_1, x_2 : A_2, \ldots, x_{k-1} : A_{k-1}$, i.e.

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

We say that a context $x_1 : A_1,\ \ldots,\ x_n : A_n$ **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.*

In the special case where $n = 0$, the list $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$ is empty, which satisfies the well-formedness condition vacuously. In other words, the **empty context** is well-formed. A well-formed type in the empty context is also called a **closed type**, and a well-formed term of a closed type is called a **closed term**.

When B is a type in context $\Gamma, x : A$, we also say that B is a **family of types** over A (in context Γ).

1.2 Renaming variables

In some situations one might want to change the name of a variable in a context. This is allowed, provided that the new variable does not occur anywhere else in the context, so that also after renaming no variable is declared more than once. The inference rules that rename a variable are sometimes called α-conversion rules.

If we are given a type A in context Γ, then for any type B in context $\Gamma, x : A, \Delta$ we can form the type $B[x'/x]$ in context $\Gamma, x' : A, \Delta[x'/x]$, where $B[x'/x]$ is an abbreviation for

$$B(x_1, \ldots, x_{n-1}, x', x_{n+1}, \ldots, x_{n+m-1})$$

This definition of **renaming** the variable x by x' is understood to be recursive over the length of Δ. The first variable renaming rule postulates that the renaming of a variable preserves well-formedness of types:

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

Similarly we obtain for any term $b : B$ in context $\Gamma, x : A, \Delta$ a term $b[x'/x] : B[x'/x]$, and there is a variable renaming rule postulating that the renaming of a variable preserves the well-formedness of terms. In fact, there is variable renaming rule for each of the primitive judgments. To avoid having to state essentially the same rule four times in a row, we postulate the four variable renaming rules all at once using a *generic judgment* $\mathcal{J}$.

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

where $\mathcal{J}$ may be a typing judgment, a judgment of equality of types, a term judgment, or a judgment of equality of terms. We will use generic judgments extensively to postulate the rest of the rules of dependent type theory.

1.3 Inference rules governing judgmental equality

Both on types and on terms, we postulate that judgmental equality is an equivalence relation. That is, we provide inference rules for the reflexivity, symmetry and transitivity of both kinds of judgmental equality:

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

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

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. We state this with a generic judgment $\mathcal{J}$

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

An analogous *term conversion rule* stated in Exercise 1.1, converting the type of a term to a judgmentally equal type, is derivable.

1.4 Structural rules of type theory

We complete the specification of dependent type theory by postulating rules for *weakening* and *substitution*, and the *variable rule*:

(i) 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.

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

This process of expanding the context by a fresh variable of type A is called **weakening (by A)**. The type family $W_A(B)$ over A is also called the **constant family** B, or the **trivial family** B.

(ii) If we are given a type A in context Γ, then x is a well-formed term of type A in context $\Gamma, x : A$.

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

This is called the **variable rule**. It provides an *identity function* on the type A in context Γ.

(iii) If we are given a term $a : A$ in context Γ, then for any type B in context $\Gamma, x : A, \Delta$ we can form the type $B[a/x]$ in context $\Gamma, \Delta[a/x]$, where $B[a/x]$ is an abbreviation for

$$B(x_1, \ldots, x_{n-1}, a(x_1, \ldots, x_{n-1}), x_{n+1}, \ldots, x_{n+m-1})$$

This definition of substituting a for x is understood to be recursive over the length of Δ. Similarly we obtain for any term $b : B$ in context $\Gamma, x : A, \Delta$ a term $b[a/x] : B[a/x]$. The **substitution rule** asserts that substitution preserves well-formedness and judgmental equality of types and terms:

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

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

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

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. Often we write $B(a)$ for $B[a/x]$.

Example 1.4.1. To give an example of how the deductive system works, we give a deduction for the **interchange rule**

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

In other words, if we have two types A and B in context Γ, and we make a judgment in context $\Gamma, x : A, y : B$, then we can make that same judgment in context $\Gamma, y : B, x : A$. The derivation is as follows:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash B\ \mathrm{type}}{\Gamma, y : B \vdash y : B}\,\delta_B}{\Gamma, y : B, x : A \vdash y : B}\,W_{W_B(A)} \qquad \dfrac{\dfrac{\Gamma, x : A, y : B, \Delta \vdash \mathcal{J}}{\Gamma, x : A, y' : B, \Delta[y'/y] \vdash \mathcal{J}[y'/y]}\,y'/y}{\Gamma, y : B, x : A, y' : B, \Delta[y'/y] \vdash \mathcal{J}[y'/y]}\,W_B}{\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}}\,S_{W_A(y)}$$

Exercises

1.1 Give a derivation for the following conversion rule:

$$\dfrac{\Gamma \vdash A \equiv A'\ \mathrm{type} \qquad \Gamma \vdash a : A}{\Gamma \vdash a : A'}$$

Lecture 2

Dependent function types and the natural numbers

2.1 Dependent function types

Dependent function types are formed from a type A and a type family B over A, i.e. the Π**-formation rule** is as follows:

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

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

Furthermore, when x' is a fresh variable, i.e. which does not occur in the context $\Gamma, x : A$, we also postulate that

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

The idea of dependent function types is that their terms are functions of which the type of the output depends on the input. In other words, they consist of constructions that provide for every $x : A$ a term $b(x) : B(x)$. Dependent functions are formed from terms $b(x)$ of type $B(x)$ in context $\Gamma, x : A$, i.e. the λ**-abstraction rule** is as follows:

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

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

Furthermore, when x' is a fresh variable, we also postulate that

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

There are also rules providing a way to *use* dependent functions. This is determined by the **evaluation rule**, which 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:

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

In other words, every term of type $B(x)$ in context $\Gamma, x : A$ determines a term of type $\prod_{(x:A)} B(x)$ in context Γ, and vice versa. The λ-abstraction rule and the evaluation rule are mutual inverses: we impose the **β-rule**

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

and the **η-rule**

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

This completes the specification of dependent function types.

Remark 2.1.1. Types of dependent functions with *multiple* arguments can be obtained by iterating the Π-construction.

Remark 2.1.2. Some authors write

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

for the dependent function type $\prod_{(x:A)} B(x)$.

Remark 2.1.3. By the derivation

$$\cfrac{\cfrac{\cfrac{\Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{(x:A)} B(x) \text{ type}}\ \Pi_A}{\Gamma, f : \prod_{(x:A)} B(x) \vdash f : \prod_{(x:A)} B(x)}\ \delta}{\Gamma, f : \prod_{(x:A)} B(x), x : A \vdash f(x) : B(x)}\ ev$$

it follows that one can also evaluate variables of type $\prod_{(x:A)} B(x)$.

2.2 Function types

In the case where both A and B are types in context Γ, we may first weaken B by A, and then apply the formation rule for the dependent function type:

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

The result is the type of functions that take an argument of type A, and return a term of type B. In other words, terms of the type $\prod_{(x:A)} B$ are *ordinary* functions from A to B. We write $A \to B$ for the **type of functions** from A to B.

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

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

$$\dfrac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash f(x) : B} ev$$

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

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

Remark 2.2.1. We also use the exponent notation B^A for the function type $A \to B$. Furthermore, we maintain the convention that the $\to$ associates to the right, i.e. when we write $A \to B \to C$, we mean $A \to (B \to C)$.

Remark 2.2.2. Similar to Remark 2.1.3, we can derive

$$\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, f : B^A, x : A \vdash f(x) : B}$$

Definition 2.2.3. For any type A in context Γ, we define the **identity function** $\mathsf{id}_A : A \to A$ using the 'variable rule':

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

Definition 2.2.4. 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

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

We will write $g \circ f$ for $\mathsf{ev}(\mathsf{ev}(\mathsf{comp}, g), f)$.

Construction. We give the following derivation to define the composition operation:

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

□

Lemma 2.2.5. *Composition of functions is associative, i.e. we can derive*

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

Proof. In the following derivation we prove that composition of functions is associative.

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

□

Lemma 2.2.6. *Composition of functions satisfies the left and right unit laws, i.e. we can derive*

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

and

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

Proof. The derivation for the left unit law is

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

The right unit law is left as Exercise 2.1. □

2.3 The natural numbers

The archetypal example of an inductive type is the type of *natural numbers*. The type of **natural numbers** is defined to be a closed type $\mathbb{N}$ equipped with closed terms for a **zero term** and a **successor function**

$$0 : \mathbb{N} \qquad \text{and} \qquad S : \mathbb{N}\to\mathbb{N},$$

To prove properties about the natural numbers, we postulate an *induction principle* for $\mathbb{N}$. In dependent type theory, however, the induction principle for the natural numbers provides a way to construct *dependent functions* of types depending on the natural numbers.

The **induction principle** for $\mathbb{N}$ states that for every type P in context $\Gamma, n : \mathbb{N}$ one can infer

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

Furthermore we require that the dependent function $\mathsf{ind}_{\mathbb{N}}(P, p_0, p_S)$ behaves as expected when it is applied to 0 or a successor, i.e. with the same hypotheses as for the induction principle we postulate the **computation rules** for $\mathbb{N}$

$$\frac{\cdots}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, 0) \equiv p_0 : P(0)}\ \mathbb{N}\text{−Comp}(0)$$

$$\frac{\cdots}{\Gamma, n : \mathbb{N} \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, S(n)) \equiv p_S(n, \mathsf{ind}_{\mathbb{N}}(p_0, p_S, n)) : P(S(n))}\ \mathbb{N}\text{−Comp}(S)$$

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

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

by induction.

Informally, the definition of addition is as follows. By induction it suffices to construct a function $add_0 : \mathbb{N} \to \mathbb{N}$, and a function

$$add_S(f) : \mathbb{N} \to \mathbb{N},$$

assuming $n : \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$. The function $add_0 : \mathbb{N} \to \mathbb{N}$ is of course taken to be $\mathsf{id}_{\mathbb{N}}$, since it has to add nothing. Given a function $f : \mathbb{N} \to \mathbb{N}$ we define $add_S(f)$ to be $\lambda m.\, S(f(m))$, simply adding one to f pointwise.

The derivation for the construction of add_S looks as follows:

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathbb{N}\ \text{type}} \qquad \dfrac{\dfrac{\dfrac{}{\vdash \mathbb{N}\ \text{type}} \quad \dfrac{}{\vdash \mathbb{N}\ \text{type}}}{f : \mathbb{N}^{\mathbb{N}}, m : \mathbb{N} \vdash f(m) : \mathbb{N}} \qquad \dfrac{\dfrac{\dfrac{}{\vdash S : \mathbb{N} \to \mathbb{N}}}{n : \mathbb{N} \vdash S(n) : \mathbb{N}}}{f : \mathbb{N}^{\mathbb{N}}, m : \mathbb{N}, n : \mathbb{N} \vdash S(n) : \mathbb{N}}}{f : \mathbb{N}^{\mathbb{N}}, m : \mathbb{N} \vdash S(f(m)) : \mathbb{N}}}{n : \mathbb{N}, f : \mathbb{N}^{\mathbb{N}}, m : \mathbb{N} \vdash S(f(m)) : \mathbb{N}}}{\vdash add_S :\equiv \lambda n.\, \lambda f.\, \lambda m.\, S(f(m)) : \mathbb{N} \to \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}}$$

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

$$\frac{\dfrac{}{\vdash add_0 :\equiv \mathsf{id}_{\mathbb{N}} : \mathbb{N}^{\mathbb{N}}} \qquad \dfrac{}{\vdash add_S : \mathbb{N} \to \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}}}{\vdash \mathsf{add} : \mathsf{ind}_{\mathbb{N}}(add_0, add_S) : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}}$$

Usually we will write $n + m$ for $\mathsf{add}(n, m)$. By the computation rules we have

$$\begin{aligned} 0 + m &\equiv m \\ S(n) + m &\equiv S(n + m) \end{aligned}$$

for any $n, m : \mathbb{N}$.

Remark 2.3.1. The rules that we provided so far are not sufficient to also conclude that $n + 0 \equiv n$ and $n + S(m) \equiv S(n + m)$. However, once we have introduced the *identity type* we will nevertheless be able to *identify* $n + 0$ with n, and $n + S(m)$ with $S(n + m)$.

Exercises

2.1 Give a derivation for the right unit law of Lemma 2.2.6.

2.2 In this exercise we generalize the composition operation of non-dependent function types:

(a) Define a composition operation for dependent function types

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x) \qquad \Gamma, x : A \vdash g : \prod_{(y:B)} C(x, y)}{\Gamma \vdash g \circ f : \prod_{(x:A)} C(x, \mathsf{ev}(f, x))}$$

and show that this operation agrees with ordinary composition when it is specialized to non-dependent function types.

(b) Show that composition of dependent functions is associative.

(c) Show that composition of dependent functions satisfies the right unit law:

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma \vdash f \circ \mathsf{id}_A \equiv f : \prod_{(x:A)} B(x)}$$

(d) Show that composition of dependent functions satisfies the left unit law:

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma \vdash \mathsf{id}_B \circ f \equiv f : \prod_{(x:A)} B(x)}$$

2.3 (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

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

2.4 (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

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

2.5 (a) Define the **multiplication** operation $\mathsf{mul} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

(b) Define the **power** operation $n, m \mapsto m^n$ of type $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

(c) Define the **factorial** function $n \mapsto n!$.

2.6 Define the binary min and max functions $\mathsf{min}, \mathsf{max} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

2.7 Construct a function

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

in context Γ, for every type P in context $\Gamma, n : \mathbb{N}$, and show that the computation rules

$$\begin{aligned} \mathsf{ind}_{\mathbb{N}}(p_0, p_S, 0) &\equiv p_0 \\ \mathsf{ind}_{\mathbb{N}}(p_0, p_S, S(n)) &\equiv p_S(n, \mathsf{ind}_{\mathbb{N}}(p_0, p_S, n)) \end{aligned}$$

hold. *Note: this is more an exercise in type theoretical bookkeeping than an exercise about the natural numbers.*

Lecture 3

Inductive types and the universe

From this chapter on we will use a more informal style of reasoning. Keeping in mind that formal deductions can be given, we will reason in prose.

3.1 Inductive types

Many other types can also be specified as inductive types, similar to the natural numbers. The unit type, the empty type, and the booleans are the simplest examples of this way of defining 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 be multiple constructors, or 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 dependent type 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.

Table 3.1: Many types can be defined as inductive types.

name	type	constructors
natural numbers	$\mathbb{N}$	$0 : \mathbb{N}$ $S : \mathbb{N} \to \mathbb{N}$
empty type	$\mathbf{0}$	(no constructors)
unit type	$\mathbf{1}$	$\star : \mathbf{1}$
booleans	$\mathbf{2}$	$0_{\mathbf{2}} : \mathbf{2}$ $1_{\mathbf{2}} : \mathbf{2}$
coproduct	$A + B$	$\mathsf{inl} : A \to A + B$ $\mathsf{inr} : B \to A + B$
product	$A \times B$	$(-,-) : A \to (B \to A \times B)$
Σ-type	$\sum_{(x:A)} B(x)$	$(-,-) : \prod_{(y:A)} \big(B(y) \to \sum_{(x:A)} B(x)\big)$

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 3.1.1. 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 $\Gamma, x : \mathbf{1} \vdash P(x)$ type, there is a term

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

in context Γ 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 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 3.1.2. We define the **empty type** to be a type $\mathbf{0}$ satisfying the induction principle that for any type family $P : \mathbf{0} \to \mathsf{Type}$, there is a term

$$\mathsf{ind}_{\mathbf{0}} : \textstyle\prod_{(x:\mathbf{0})} P(x).$$

Using the empty type we can also define *negation*. The idea is that if A is false (i.e. has no terms), then from A follows everything.

Definition 3.1.3. For any type A, we define $\neg A :\equiv A \to \mathbf{0}$.

Unlike set theory, in most type theories every term has a *unique* type. Therefore we annotate the constructors of **2** with their type, to not confuse them with the terms 0 and 1 of the natural numbers.

Definition 3.1.4. We define the **booleans** to be a type **2** that comes equipped with

$$\begin{gathered} 0_{\mathbf{2}} : \mathbf{2} \\ 1_{\mathbf{2}} : \mathbf{2} \end{gathered}$$

satisfying the induction principle that for any type family $P : \mathbf{2} \to \mathsf{Type}$, there is a term

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

for which the computation rules

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

hold.

Definition 3.1.5. Let A and B be types. We define the **coproduct** $A + B$ to be a type that comes equipped with

$$\begin{gathered} \mathsf{inl} : A \to A + B \\ \mathsf{inr} : B \to A + B \end{gathered}$$

satisfying the induction principle that for any type family $P : (A + B) \to \mathsf{Type}$, there is a term

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

for which the computation rules

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

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

The coproduct of two types is sometimes also called the **disjoint sum**. When one thinks of types as propositions, then the coproduct plays the role of the disjunction. To construct a term of type $A + B$ you first have to decide whether it is of the form inl or inr, and then you construct a term of A or B accordingly. Of course, this is to be contrasted with the *double negation translation* of the disjunction, which is read as 'not neither A nor B'.

The *dependent pair type* (or Σ-type) can be thought of as a 'type indexed' disjoint sum. However, this intuition for the dependent pair type can be counterproductive once we start to do homotopy theory in type theory. It is better to think of the Σ-type as the total space of a family of types depending continuously on a base type, just like one can have a family of spaces depending continuously on a base space (i.e. a fibration).

Definition 3.1.6. Let A be a type in context Γ, and let $\Gamma, x : A \vdash B(x)$ type be a type family over A. The **dependent pair type** is defined to be the inductive type $\sum_{(x:A)} B(x)$ in context Γ equipped with a **pairing function**

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

The induction principle for $\sum_{(x:A)} B(x)$ asserts that for every type family

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

one has

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

satisfying the computation rule

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

Most of the time we write $\lambda(x,y).\, f(x,y)$ for $\mathsf{ind}_\Sigma(\lambda x.\, \lambda y.\, f(x,y))$.

Remark 3.1.7. Some authors write $(x : A) \times B(x)$ for the dependent pair type $\sum_{(x:A)} B(x)$.

Definition 3.1.8. Given a type A and a type family B over A, the **first projection map**

$$\mathsf{pr}_1 : \Big(\sum_{(x:A)} B(x)\Big) \to A$$

is defined by induction as

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

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

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

defined by induction as

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

By the computation rule we have

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

When one thinks of types as propositions, then the Σ-type has the rôle of the existential quantification.

A special case of the Σ-type occurs when the B is a type in context Γ weakened by A (i.e. B is not actually depending on A). In this case, a term of $\sum_{(x:A)} B$ is given as a pair consisting of a term of A and a term of B. Thus, $\sum_{(x:A)} B$ is the *(cartesian) product)* of A and B. Since the cartesian product is so common (just like $A \to B$ is a common special case of the dependent product), we provide its definition.

Definition 3.1.9. Let A and B be types in context Γ. 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

$$\mathsf{ind}_\times : \Big(\textstyle\prod_{(x:A)}\prod_{(y:B)} P((-,-))\Big) \to \Big(\prod_{(p:A\times B)} P(p)\Big)$$

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.

3.2 The universe

The induction principle for inductive types can be used to prove universal quantifications. However, it would also be nice if we could construct *new type families* over inductive types, using their induction principles. To be able to do this, we introduce a *universe*, a type of which the terms represent types. The idea is that the universe $\mathcal{U}$ comes equipped with a type family El, so that for each $X : \mathcal{U}$ we have an associated type $\mathrm{El}(X)$, the type of *elements* of X.

We assume there is a closed type $\mathcal{U}$ called the **universe**, and a type family El over $\mathcal{U}$ called the **universal family**.

$$\frac{}{\vdash \mathcal{U}\ \mathrm{type}} \qquad\qquad \frac{}{X : \mathcal{U} \vdash \mathrm{El}(X)\ \mathrm{type}}$$

We postulate that the universe is closed under the type constructors, by the following rules:

(i) The universe is closed under Π-types

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma \vdash B : \mathrm{El}(A) \to \mathcal{U}}{\Gamma \vdash \check{\Pi}(A, B) : \mathcal{U}}$$

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma \vdash B : \mathrm{El}(A) \to \mathcal{U}}{\Gamma \vdash \mathrm{El}(\check{\Pi}(A, B)) \equiv \prod_{(x:\mathrm{El}(A))} \mathrm{El}(B(x))\ \mathrm{type}}$$

(ii) The type of natural numbers is in the universe

$$\frac{}{\vdash \check{\mathbb{N}} : \mathcal{U}}$$

$$\frac{}{\vdash \mathrm{El}(\check{\mathbb{N}}) \equiv \mathbb{N}\ \mathrm{type}}$$

(iii) Similarly we postulate that the universe contains the empty type, the unit type, the booleans, coproducts, products, and Σ-types. These closure properties of the universe are given concisely in Table 3.2.

Definition 3.2.1. We say that a type A in context Γ is **small** if it occurs in the universe, i.e. if there is a term $\check{A} : \mathcal{U}$ in context Γ such that $\Gamma \vdash \mathrm{El}(\check{A}) \equiv A$ type.

In particular, if A is a small type in context Γ and B is a small type in context $\Gamma, x : A$, then $\prod_{(x:A)} B(x)$ is again a small type in context Γ.

Table 3.2: Closure properties of the universe

Premises	Type encoding in $\mathcal{U}$	Type of elements $\mathrm{El}(-)$
$A : \mathcal{U}, B : \mathrm{El}(A) \to \mathcal{U}$	$\check{\Pi}(A, B)$	$\prod_{(x:\mathrm{El}(A))} \mathrm{El}(B(x))$
	$\check{\mathbb{N}}$	$\mathbb{N}$
	$\check{\mathbf{0}}$	$\mathbf{0}$
	$\check{\mathbf{1}}$	$\mathbf{1}$
	$\check{\mathbf{2}}$	$\mathbf{2}$
$A, B : \mathcal{U}$	$A \check{+} B$	$\mathrm{El}(A) + \mathrm{El}(B)$
$A, B : \mathcal{U}$	$A \check{\times} B$	$\mathrm{El}(A) \times \mathrm{El}(B)$
$A : \mathcal{U}, B : \mathrm{El}(A) \to \mathcal{U}$	$\check{\Sigma}(A, B)$	$\sum_{(x:\mathrm{El}(A))} \mathrm{El}(B(x))$

Definition 3.2.2. Let A be a type in context Γ. A **family of small types** over A is defined to be a map

$$B : A \to \mathcal{U}$$

Remark 3.2.3. If A is small, we usually write simply A for $\check{A}$ and also A for $\mathrm{El}(\check{A})$. In other words, by $A : \mathcal{U}$ we mean that A is a small type.

Example 3.2.4. One important way to use the universe is to define types of **structured types**. We give some examples:

(i) The type of small **pointed types** is defined as

$$\mathcal{U}_* :\equiv \textstyle\sum_{(A:\mathcal{U})} A,$$

(ii) The type of small **graphs** is defined as the type

$$\mathsf{Gph}_{\mathcal{U}} :\equiv \textstyle\sum_{(A:\mathcal{U})} A \to (A \to \mathcal{U}),$$

(iii) The type of small **reflexive graphs** is defined as the type

$$\mathsf{rGph}_{\mathcal{U}} :\equiv \textstyle\sum_{(A:\mathcal{U})} \sum_{(R:A\to(A\to\mathcal{U}))} \prod_{(a:A)} R(a, a).$$

Once we have introduced the *identity types* we will also be able to state the types of groups, rings, and many other structured types. However, when doing so one has to be cautious to make sure that the underlying type is in the level of sets, in the hierarchy of homotopical complexity of types.

Another important way to use the universe is to *define* new type families by induction. For example, we can define the finite types as family over the natural numbers.

Definition 3.2.5. We define the type family $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$ of finite types by induction on $\mathbb{N}$, taking

$$\begin{aligned} \mathsf{Fin}(0) &:\equiv \mathbf{0} \\ \mathsf{Fin}(n+1) &:\equiv \mathsf{Fin}(n) + \mathbf{1} \end{aligned}$$

A second example of this kind is the notion of *observational equality* on the natural numbers.

Definition 3.2.6. We define the **observational equality** on $\mathbb{N}$ as binary relation $\mathrm{Eq}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ satisfying

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

Construction. We define $\mathrm{Eq}_{\mathbb{N}}$ 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 $E_{0S}(n,X,m) :\equiv \mathbf{0}$. The resulting family E_0 satisfies

$$\begin{aligned} E_0(0) &\equiv \mathbf{1} \\ E_0(S(n)) &\equiv \mathbf{0}. \end{aligned}$$

We define E_S by induction, taking $E_{S0} :\equiv \mathbf{0}$ and $E_{S0}(n,X,m) :\equiv X(m)$. The resulting family E_S satisfies

$$\begin{aligned} E_S(n,X,0) &\equiv \mathbf{0} \\ E_S(n,X,S(m)) &\equiv X(m) \end{aligned}$$

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

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

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

3.3 The type of integers

Definition 3.3.1. We define the **integers** to be the type $\mathbb{Z} :\equiv \mathbb{N} + (\mathbf{1} + \mathbb{N})$, and we write

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

In the following lemma we derive an alternative induction principle for $\mathbb{Z}$, which makes it easier to make definitions.

Lemma 3.3.2. *For any* $\Gamma, k : \mathbb{Z} \vdash P(k)$ type *we have*

$$\frac{\begin{array}{l}
\Gamma \vdash p_{-1} : P(-1) \\
\Gamma \vdash p_{-S} : \prod_{(n:\mathbb{N})} P(\mathsf{neg}(n)) \to P(\mathsf{neg}(S(n))) \\
\Gamma \vdash p_0 : P(0) \\
\Gamma \vdash p_1 : P(1) \\
\Gamma \vdash p_{-S} : \prod_{(n:\mathbb{N})} P(\mathsf{pos}(n)) \to P(\mathsf{pos}(S(n)))
\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{Z}}(p_{-1}, p_{-S}, p_0, p_1, p_S) : \prod_{(k:\mathbb{Z})} P(k)}$$

The term $\mathsf{ind}_{\mathbb{Z}}(p_{-1}, p_{-S}, p_0, p_1, p_S)$ *furthermore satisfies the following computation rules:*

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

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

Definition 3.3.3. We define the **successor function** on the integers $S_{\mathbb{Z}} : \mathbb{Z} \to \mathbb{Z}$.

Construction. We apply the induction principle of Lemma 3.3.2, taking

$$S_{\mathbb{Z}}(-1) :\equiv 0$$

$$\begin{aligned} S_{\mathbb{Z}}(\mathsf{neg}(S(n))) &:\equiv \mathsf{neg}(n) \\ S_{\mathbb{Z}}(0) &:\equiv 1 \\ S_{\mathbb{Z}}(1) &:\equiv \mathsf{pos}(S(1)) \\ S_{\mathbb{Z}}(\mathsf{pos}(S(n))) &:\equiv \mathsf{pos}(S(S(n))). \end{aligned}$$

□

Exercises

3.1 For any type A, show that $(A + \neg A) \to (\neg\neg A \to A)$.

3.2 Construct a function

$$\check{\Pi} : \prod_{(A:\mathcal{U})}(\mathrm{El}(A) \to \mathcal{U}) \to \mathcal{U}$$

such that

$$\mathrm{El}(\check{\Pi}(A, B)) \equiv \prod_{(x:\mathrm{El}(A))}\mathrm{El}(B(x))$$

holds for every $A : \mathcal{U}$ and $B : \mathrm{El}(A) \to \mathcal{U}$.

A similar exercise can be posed for Σ and $+$ (and for $\to$ and $\times$ as special cases of Π and Σ).

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

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

3.4 Let R be a reflexive binary relation on $\mathbb{N}$, i.e. R is of type $\mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ and comes equipped with a term $\rho : \prod_{(n:\mathbb{N})} R(n, n)$. Show that

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

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

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

Hint: to get the inductive step going the induction hypothesis has to be strong enough. Construct by double induction a term of type

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

and pull out the universal quantification over $f : \mathbb{N} \to \mathbb{N}$ by Exercise 2.4.

3.6 (a) Define the **order relations** $\leq$ and $<$ on $\mathbb{N}$.
(b) Show that $\leq$ is reflexive and that $<$ is **anti-reflexive**, i.e. that $\neg(n < n)$.
(c) Show that both $\leq$ and $<$ are transitive, and that $n < S(n)$.

3.7 Use the observational equality of the natural numbers to define the **divisibility relation** $d \mid n$.

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

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

3.9 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 every $\Gamma, t : \mathbf{1}+\mathbf{1} \vdash P(t)$ type there is a dependent function of type

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

satisfying

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

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

3.10 (a) Define the order relations $\leq$ and $<$ on and $\mathbb{Z}$.
(b) For $k : \mathbb{Z}$, consider the type $\mathbb{Z}_{\geq k} :\equiv \sum_{(n:\mathbb{Z})} n \geq k$. Construct

$$b_k : \mathbb{Z}_{\geq k}$$
$$s_k : \mathbb{Z}_{\geq k} \to \mathbb{Z}_{\geq k},$$

and show that $\mathbb{Z}_{\geq k}$ satisfies the induction principle of the natural numbers:

$$\mathsf{ind}_{\mathbb{Z}_{\geq k}} : P(b_k) \to \Big(\prod_{(n:\mathbb{Z}_{\geq k})} P(n) \to P(s_k(n))\Big) \to \Big(\prod_{(n:\mathbb{Z}_{\geq k})} P(n)\Big)$$

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

3.12 Define operations $k, l \mapsto k + l : \mathbb{Z} \to \mathbb{Z} \to \mathbb{Z}$ and $k \mapsto -k : \mathbb{Z} \to \mathbb{Z}$.

Lecture 4

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 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 groupoid-like 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!

Table 4.1: The homotopy interpretation

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

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 4.1. 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.

4.1 The inductive definition of identity types

Let A be a type in context Γ. The **identity type** of A at $a : A$ is the inductive type family

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

with constructor

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

The induction principle that for any type family

$$\Gamma, x : A, y : A, \alpha : x =_A y \vdash P(x, y, \alpha) \text{ type}$$

there is a term

$$\mathsf{ind}_{x=} : P(x, x, \mathsf{refl}_x) \to \textstyle\prod_{(y:A)}\prod_{(\alpha : x =_A y)} P(x, y, \alpha)$$

in context $\Gamma, x : A$, satisfying the computation rule

$$\mathsf{ind}_{x=}(p, x, \mathsf{refl}_x) \equiv p.$$

A term of type $x =_A y$ is also called an **identification** of x with y, and sometimes it is called a **path** from x to y. 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.

We also assume that the universe $\mathcal{U}$ is closed under identity types, i.e. that there is a map

$$\check{\mathsf{Id}} : \prod_{(A:\mathcal{U})} \mathrm{El}(A) \to \mathrm{El}(A) \to \mathcal{U}$$

satisfying

$$\mathrm{El}(\check{\mathsf{Id}}(A,x,y)) \equiv x =_{\mathrm{El}(A)} y.$$

In the following lemma we show that the identity type on A is contained in any reflexive relation on A.

Lemma 4.1.1. *Let* $\Gamma, x : A, y : A \vdash R(x,y)$ type*, and suppose that* R *is reflexive in the sense that there is a term*

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

Then there is a term of type

$$\prod_{(y:A)} (x =_A y) \to R(x,y)$$

in context $\Gamma, x : A$*.*

Construction. By weakening the reflexive relation R we obtain

$$\Gamma, x : A, y : A, \alpha : x =_A y \vdash R(x,y) \text{ type},$$

on which the induction principle is applicable. Thus we see that by the induction principle for identity types we have a term

$$\mathsf{ind}_{x=} : R(x,x) \to \prod_{(y:A)} (x =_A y) \to R(x,y)$$

so it suffices to construct a term of type $R(x,x)$, which we have by reflexivity of R. □

4.2 The groupoid structure of types

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

Furthermore, we observe that we can iteratively take identity types, i.e. we can take identity types of identity types,

$$p =_{(x =_A y)} q,$$

and so on. In other words, for any two identifications $p, q : x =_A y$, there is a type of identifications of p with y. One way to think about this is that the identifications $p, q : x =_A y$ are paths in the type (space) A, and an identification of p with q is a *higher path* from p to q, i.e. a *homotopy*.

Using the observation that identity types can be iterated we show that concatenation is *associative*, satisfies the left and right *unit laws*, and satisfies the left and right *inverse laws*. These are the **groupoid operations** on the identity type.

Definition 4.2.1. 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 $\mathsf{concat}(p, q)$. Also, we will *associate to the right*, i.e. by $p \cdot q \cdot r$ we mean $p \cdot (q \cdot r)$.

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

$$\lambda x.\, \mathsf{rec}_{x=}(\lambda z.\, \mathsf{id}_{(x=z)}) : \prod_{(x,y:A)} (x = y) \to \prod_{(z:A)} (y = z) \to (x = z).$$

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 2.4. □

Definition 4.2.2. 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 $\mathsf{inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_x$. □

Definition 4.2.3. Let A be a type. We define the **associativity operation**, which assigns to each $p : x = y$, $q : y = z$, and $r : z = w$ the **associator**

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

Construction. By identification elimination it suffices to show that

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

Let $q : x = z$ and $r : z = w$. Note that by the computation rule $\mathsf{refl}_x \cdot q \equiv q$, so $(\mathsf{refl}_x \cdot q) \cdot r \equiv q \cdot r$. Similarly we have $\mathsf{refl}_x \cdot (q \cdot r) \equiv q \cdot r$. Therefore we can simply take $\mathsf{refl}_{q \cdot r}$. □

Definition 4.2.4. Let A be a type. We define the left and right **unit operations**, which assigns to each $p : x = y$ the terms

$$\begin{aligned}\mathsf{left\_unit}(p) &: \mathsf{refl}_x \cdot p = p\\ \mathsf{right\_unit}(p) &: p \cdot \mathsf{refl}_y = p,\end{aligned}$$

respectively.

Construction. By identification elimination it suffices to construct

$$\begin{aligned}\mathsf{left\_unit}(\mathsf{refl}_x) &: \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x\\ \mathsf{right\_unit}(\mathsf{refl}_x) &: \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x.\end{aligned}$$

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

Definition 4.2.5. Let A be a type. We define left and right **inverse operations**

$$\begin{aligned}\mathsf{left\_inv}(p) &: p^{-1} \cdot p = \mathsf{refl}_y\\ \mathsf{right\_inv}(p) &: p \cdot p^{-1} = \mathsf{refl}_x.\end{aligned}$$

Construction. By identification elimination it suffices to construct

$$\begin{aligned}\mathsf{left\_inv}(\mathsf{refl}_x) &: \mathsf{refl}_x^{-1} \cdot \mathsf{refl}_x = \mathsf{refl}_x\\ \mathsf{right\_inv}(\mathsf{refl}_x) &: \mathsf{refl}_x \cdot \mathsf{refl}_x^{-1} = \mathsf{refl}_x.\end{aligned}$$

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 $\mathsf{left\_inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}$. 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 $\mathsf{right\_inv}(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}$. □

4.3 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 4.3.1. Let $f : A \to B$ be a map. We define the **action on paths** of f as an operation

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

Moreover, there are operations

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

Construction. First we define ap_f by identity elimination, taking

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

Next, we construct $\mathsf{ap.idfun}_A$ by identity elimination, taking

$$\mathsf{ap.idfun}_A(\mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}.$$

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

$$\mathsf{ap.comp}(f,g,\mathsf{refl}_x) :\equiv \mathsf{refl}_{g(f(x))}.$$ □

Definition 4.3.2. Let $f : A \to B$ be a map. Then there are identifications

$$\begin{aligned} \mathsf{ap.refl}(f,x) &: \mathsf{ap}_f(\mathsf{refl}_x) = \mathsf{refl}_f(x) \\ \mathsf{ap.inv}(f,p) &: \mathsf{ap}_f\left(p^{-1}\right) = \mathsf{ap}_f(p)^{-1} \\ \mathsf{ap.concat}(f,p,q) &: \mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q) \end{aligned}$$

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

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

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

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

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

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

$$\mathsf{ap.concat}(f,\mathsf{refl}_x,q) :\equiv \mathsf{refl}_{\mathsf{ap}_f(q)}.$$ □

4.4 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 4.4.1. 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 4.4.2. 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

4.1 Let B be a family over a type A. Construct for any two identifications $p : x =_A y$ and $q : y =_A z$, and any $b : B(x)$ an identification

$$\mathsf{tr}_B(q, \mathsf{tr}_B(p, x)) = \mathsf{tr}_B(p \cdot q, x).$$

4.2 Let $p : x = y$ and $q : y = z$. Construct an identification

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

4.3 Consider two types A and B, and let $p : x = y$ in A, and $b : B$.

(a) Construct an identification

$$\mathsf{tr\_triv}(p, b) : \mathsf{tr}_{W_A(B)}(p, b) = b$$

where $W_A(B)$ is the family B weakened by A.

(b) Construct for any $f : A \to B$, an identification

$$\mathsf{apd}_f(p) = \mathsf{tr\_triv}(p, f(x)) \cdot \mathsf{ap}_f(p),$$

witnessing that the triangle

$$\begin{tikzcd}
\mathsf{tr}_{W_A(B)}(p, f(x)) \arrow[rr, equal, "\mathsf{tr\_triv}(p,f(x))"] \arrow[dr, equal, "\mathsf{apd}_f(p)"'] & & f(x) \\
& f(y) \arrow[ur, equal, "\mathsf{ap}_f(p)"'] &
\end{tikzcd}$$

commutes.

4.4 Let $f : A \to B$ be a map, and consider $p : x = y$ in A.

(a) Construct for any $q : f(x) = b$ in B an identification

$$\mathsf{tr\_ap}(p, q) : \mathsf{tr}_{f(-)=b}(p, q) = \mathsf{ap}_f\,(p)^{-1} \cdot q.$$

(b) Similarly, construct for any $q' : b = f(x)$ in B an identification

$$\mathsf{tr\_ap}'(p, q') : \mathsf{tr}_{b=f(-)}(p, q) = q \cdot \mathsf{ap}_f\,(p)\,.$$

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

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

4.6 Let A be a type, and let B be a type family over A. Construct the **path lifting** operation

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

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.

4.7 Show that the operations of addition and multiplication on the natural numbers satisfy the following laws:

$$\begin{aligned} m+(n+k) &= (m+n)+k & m\cdot(n\cdot k) &= (m\cdot n)\cdot k \\ m+0 &= m & m\cdot 1 &= m \\ 0+m &= m & 1\cdot m &= m \\ m+n &= n+m & m\cdot n &= n\cdot m \\ & & m\cdot(n+k) &= m\cdot n+m\cdot k. \end{aligned}$$

Lecture 5

Equivalences

5.1 Homotopies

In homotopy type theory, a homotopy is just a pointwise equality between two functions f and g.

Definition 5.1.1. Let $f, g : \prod_{(x:A)} P(x)$ be two dependent functions. The type of **homotopies** from f to g is defined as

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

Since we formulated homotopies using dependent functions, we may also consider homotopies *between* homotopies, and further homotopies between those higher homotopies. Explicitly, if $H, K : f \sim g$, then the type $H \sim K$ of homotopies is just the type

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

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

Definition 5.1.2. For any dependent type $B : A \to \mathsf{Type}$ there are operations

$$\begin{aligned}
&\mathsf{htpy.refl} && : \textstyle\prod_{(f:\prod_{(x:A)} B(x))} f \sim f \\
&\mathsf{htpy.inv} && : \textstyle\prod_{\{f,g:\prod_{(x:A)} B(x)\}} (f \sim g) \to (g \sim f) \\
&\mathsf{htpy.concat} && : \textstyle\prod_{\{f,g,h:\prod_{(x:A)} B(x)\}} (f \sim g) \to (g \sim h) \to (f \sim h).
\end{aligned}$$

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

Furthermore, we define

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

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

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

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

$$\begin{aligned}
\mathsf{htpy.assoc}(H,K,L) &:\equiv \lambda x.\, \mathsf{assoc}(H(x), K(x), L(x)) \\
\mathsf{htpy.left\_unit}(H) &:\equiv \lambda x.\, \mathsf{left\_unit}(H(x)) \\
\mathsf{htpy.right\_unit}(H) &:\equiv \lambda x.\, \mathsf{right\_unit}(H(x)) \\
\mathsf{htpy.left\_inv}(H) &:\equiv \lambda x.\, \mathsf{left\_inv}(H(x)) \\
\mathsf{htpy.right\_inv}(H) &:\equiv \lambda x.\, \mathsf{right\_inv}(H(x)).
\end{aligned}$$

□

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

Definition 5.1.3. 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

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

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

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

5.2 Bi-invertible maps

Definition 5.2.1. 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 f 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

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

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

Remark 5.2.2. 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

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

Clearly, if f is **invertible** 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

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

Definition 5.2.3. Any equivalence can be given the structure of an invertible map.

Construction. 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 $K :\equiv (Hg)^{-1} \cdot hG.$ from which we obtain a homotopy $K : g \sim h$. This allows us 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 5.2.4. *The inverse of an equivalence is again an equivalence.*

Proof. Let $f : A \to B$ be an equivalence. By Definition 5.2.3 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. □

Theorem 5.2.5. *For any type A, the identity function id_A is an equivalence.*

Proof. The identity function is trivially its own section and its own retraction. □

Example 5.2.6. Let A and B be types in context Γ. For any $\Gamma, x : A, y : B \vdash C(x,y)$ type, the map

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

is an equivalence by Exercise 2.4.

5.3 The identity type of a Σ-type

In the following theorem we characterize the identity type of a Σ-type as a Σ-type of identity types.

Theorem 5.3.1. *Let B be a type family over A, let $s : \sum_{(x:A)} B(x)$, and consider the dependent function*

$$\mathsf{pair\_eq}_s : \textstyle\prod_{(t:\sum_{(x:A)} B(x))}(s = t) \to \sum_{(\alpha:\mathsf{pr}_1(s)=\mathsf{pr}_1(t))}\mathsf{tr}_B(\alpha, \mathsf{pr}_2(s)) = \mathsf{pr}_2(t)$$

defined as $\mathsf{pair\_eq}_s :\equiv \mathsf{ind}_{s=}(\mathsf{refl}_{\mathsf{pr}_1(s)}, \mathsf{refl}_{\mathsf{pr}_2(s)})$*. Then* $\mathsf{pair\_eq}_{s,t}$ *is an equivalence for every* $t : \sum_{(x:A)} B(x)$*.*

Proof. The maps in the converse direction

$$\mathsf{eq\_pair}_{s,t} : \Big(\textstyle\sum_{(p:\mathsf{pr}_1(s)=\mathsf{pr}_1(t))}\mathsf{tr}_B(p, \mathsf{pr}_2(s)) = \mathsf{pr}_2(t)\Big) \to (s = t)$$

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

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

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 $\mathsf{refl}_{(x,y)}$.

Next, we must show that $\mathsf{eq\_pair}_{s,t}$ is a section of $\mathsf{pair\_eq}_{s,t}$. In other words, we must construct an identification

$$\mathsf{pair\_eq}(\mathsf{eq\_pair}(p,q)) = (p,q)$$

for each $(p,q) : \sum_{(p:x=x')} \mathsf{tr}_B(p,y) = y'$. We proceed by path induction on p, followed by path induction on q. Our goal is now to construct a term of type

$$\mathsf{pair\_eq}(\mathsf{eq\_pair}(\mathsf{refl}_x, \mathsf{refl}_y)) = (\mathsf{refl}_x, \mathsf{refl}_y)$$

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

To complete the proof, we must show that $\mathsf{eq\_pair}_{s,t}$ is a retraction of $\mathsf{pair\_eq}_{s,t}$. In other words, we must construct an identification

$$\mathsf{eq\_pair}(\mathsf{pair\_eq}(p)) = p$$

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

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

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

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

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

Corollary 5.3.2. *Let B be a type family over A, and let $(x,y),(x',y') : \sum_{(x:A)} B(x)$. Then the map*

$$\mathsf{pair\_eq}_{(x,y),(x',y')} : ((x,y) = (x',y')) \to \left(\textstyle\sum_{(p:x=x')} \mathsf{tr}_B(p,y) = y'\right)$$

is an equivalence.

Exercises

5.1 Show that for any term $a : A$ the functions

$$\begin{aligned} \mathsf{ind}_{\mathbf{1}}(a) &: \mathbf{1} \to A \\ \mathsf{const}_a &: \mathbf{1} \to A \end{aligned}$$

are homotopic.

5.2 Let A and B be types, and consider the constant map $\mathsf{const}_b : A \to B$ for some $b : B$. Construct a homotopy

$$\mathsf{ap}_{\mathsf{const}_b}(x, y) \sim \mathsf{const}_{\mathsf{refl}_b}$$

for any $x, y : A$.

5.3 Show that $\mathsf{inv} : (x = y) \to (y = x)$, $\mathsf{concat}(p) : (y = z) \to (x = z)$, and $\mathsf{tr}_B(p) : B(x) \to B(y)$ are equivalences. What are their inverses?

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

$$\mathsf{is\_equiv}(f) \leftrightarrow \mathsf{is\_equiv}(g).$$

5.5 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. Show that f has a section if and only if g has a section.

(b) Suppose that the map g has a retraction. Show 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

$$f, \qquad g, \qquad g \circ f$$

are equivalences, then so is the third.

5.6 Show that the negation function on the booleans is an equivalence. Also show that for any function $f : \mathbf{2} \to \mathbf{2}$, if $f(0_{\mathbf{2}}) = f(1_{\mathbf{2}})$ then f is *not* an equivalence.

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

5.8 Construct a equivalences $A + B \simeq B + A$ and $A \times B \simeq B \times A$.

5.9 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 $i(x) = i(y)$.

5.10 Let $B : A \to \mathsf{Type}$, and let C be a family over $x : A, y : B(x)$. Construct an equivalence

$$\Sigma.\mathsf{assoc} : \Big(\textstyle\sum_{(p:\sum_{(x:A)} B(x))} C(\mathsf{pr}_1(p), \mathsf{pr}_2(p))\Big) \simeq \Big(\textstyle\sum_{(x:A)}\sum_{(y:B(x))} C(x, y)\Big).$$

5.11 Let A and B be types, and let C be a family over $x : A, y : B$. Construct an equivalence

$$\mathsf{\Sigma.swap} : \Big(\textstyle\sum_{(x:A)}\sum_{(y:B)} C(x,y)\Big) \simeq \Big(\textstyle\sum_{(y:B)}\sum_{(x:A)} C(x,y)\Big).$$

5.12 To define all the proof terms involved in showing that the integers form an abelian group is fairly involved. We suggest to show first that $\mathbb{Z}$ is a retract of $\mathbb{N} \times \mathbb{N}$ in a way that is compatible with addition.

(a) Define the map $\mathbb{Z} \to \mathbb{N} \times \mathbb{N}$ by

$$\begin{aligned} \mathsf{neg}(k) &\mapsto (0,k) \\ 0 &\mapsto (0,0) \\ \mathsf{pos}(k) &\mapsto (k,0) \end{aligned}$$

for $k : \mathbb{N}$. Construct a retraction $r : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ of this map, in such a way that

$$r((m+m', n+n')) = r(m,n) + r(m',n')$$

for any $m, n : \mathbb{N}$.

(b) Use this retraction to show that the operations $k, l \mapsto k + l$ and $k \mapsto -k$ on the integers defined in Exercise 3.12 satisfy the group laws:

$$\begin{aligned} k + (l + m) &= (k + l) + m \\ k + 0 &= k \\ 0 + k &= k \\ k + (-k) &= 0 \\ (-k) + k &= 0, \end{aligned}$$

and that $k + l = l + k$, making $(\mathbb{Z}, 0, +, -)$ into an abelian group.

Lecture 6

Contractible types and contractible maps

6.1 Contractible types

Theorem 6.1.1. *Let A be a type. The following are equivalent:*

*(i) A is **contractible**: there is a term of type*

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

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

*(ii) A comes equipped with a term $a : A$, and satisfies **singleton induction**: for every type family B over A, the map*

$$\Big(\textstyle\prod_{(x:A)} B(x)\Big) \to B(a)$$

given by $f \mapsto f(a)$ has a section. In other words, we have a function and a homotopy

$$\begin{aligned} \mathsf{sing\_ind}_{A,a} &: B(a) \to \textstyle\prod_{(x:A)} B(x) \\ \mathsf{sing\_comp}_{A,a} &: \textstyle\prod_{(b:B(a))} \mathsf{sing\_ind}_{A,a}(b, a) = b. \end{aligned}$$

Remark 6.1.2. 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.

Also note that the 'computation rule' in the singleton induction for A is stated using an *identification* rather than as a judgmental equality.

Proof of Theorem 6.1.1. Suppose A is contractible with center of contraction c and contraction C. First we observe that, without loss of generality, we may assume that C comes equipped with an identification $p : C(c) = \mathsf{refl}_c$. To see this, note that we can always define a new contraction C' by

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

which satisfies the requirement by the left inverse law, constructed in Definition 4.2.5.

To show that A satisfies singleton induction let B be a type family over A equipped with $b : B(a)$. We define $\mathsf{sing\_ind}(b) : \prod_{(x:A)} B(x)$ by $\lambda x.\, \mathsf{tr}_B(C(x), b)$. To see that $\mathsf{sing\_ind}(c) = b$ note that we have

$$\mathsf{tr}_B(C(c), b) \xrightarrow{\mathsf{ap}_{\lambda\omega.\,\mathsf{tr}_B(\omega,b)}(p)} \mathsf{tr}_B(\mathsf{refl}_c, b) \xrightarrow{\mathsf{refl}_b} b.$$

This 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

$$\mathsf{sing\_ind}_{A,a} : a = a \to \prod_{(x:A)} a = x.$$

Therefore $\mathsf{sing\_ind}_{A,a}(\mathsf{refl}_a)$ is a contraction. □

Example 6.1.3. By definition the unit type $\mathbf{1}$ satisfies singleton induction, so it is contractible.

Theorem 6.1.4. *For any $x : A$, the type*

$$\sum_{(y:A)} x = y$$

is contractible.

In the following proof we stress the analogy of path induction with singleton elimination, as we did in class. An alternative proof can be found in Lemma 3.11.8 of the HoTT book [2].

Proof. We will prove the statement by showing that $\sum_{(y:A)} x = y$ satisfies singleton induction, applying Theorem 6.1.1.

We have the term $(x, \mathsf{refl}_x) : \sum_{(y:A)} x = y$. Thus we need to show that for any type family B over $\sum_{(x:A)} x = y$, the map

$$\Big(\prod_{(t:\sum_{(x:A)} x=y)} B(t)\Big) \to B((x, \mathsf{refl}_x))$$

has a section. Note that we have the composite of maps

$$B((x, \mathsf{refl}_x)) \xrightarrow{\mathsf{ind}_{x=}} \prod_{(y:A)} \prod_{(p:x=y)} B((y,p)) \xrightarrow{\mathsf{ind}_{\Sigma}} \prod_{(t:\sum_{(y:A)} x=y)} B(t)$$

which we take as our definition of $\mathsf{sing\_ind}$. Moreover, by the computation rules we have

$$\mathsf{ind}_{\Sigma}(\mathsf{ind}_{x=}(b), (x, \mathsf{refl}_x)) \equiv b.$$

Thus, for $\mathsf{sing\_comp}$ we simply take $\lambda b.\, \mathsf{refl}_b$. □

6.2 Contractible maps

Definition 6.2.1. Let $f : A \to B$ be a function, and let $b : B$. The **fiber** of f at b is defined to be the type

$$\mathsf{fib}_f(b) :\equiv \sum_{(a:A)} f(a) = b.$$

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.

Definition 6.2.2. We say that a function $f : A \to B$ is **contractible** if there is a term of type

$$\mathsf{is\_contr}(f) :\equiv \prod_{(b:B)} \mathsf{is\_contr}(\mathsf{fib}_f(b)).$$

Theorem 6.2.3. *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

$$\lambda y.\, (g(y), G(y)) : \prod_{(y:B)} \mathsf{fib}_f(y).$$

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. □

6.3 Equivalences are contractible maps

In this section we will show the converse to Theorem 6.2.3, that equivalences are contractible maps. Before we do so, we will establish some useful constructions on homotopies and section-retraction pairs.

Definition 6.3.1. Let $f, g : A \to B$ be functions, and consider $H : f \sim g$ and $p : x = y$ in A. We define identification

$$\mathsf{htpy\_nat}(H, p) :\equiv \mathsf{ind}_{x=}(\mathsf{right\_unit}(H(x)), p) : H(x) \cdot \mathsf{ap}_g(p) = \mathsf{ap}_f(p) \cdot H(y)$$

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.

Definition 6.3.2. 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 $H(f(x)) = \mathsf{ap}_f(H(x))$. □

Theorem 6.3.3. *Any equivalence is a contractible map.*

Proof. Since every equivalence has the structure of an invertible map by Definition 5.2.3, it suffices to show that any invertible map is contractible.

Let $f : A \to B$ be a map, with $g : B \to A$, $G : f \circ g \sim \mathsf{id}_B$, and $H : g \circ f \sim \mathsf{id}_A$. We have for any $y : B$ the term $(g(y), G(y)) : \mathsf{fib}_f(y)$. However, as our center of contraction we take $(g(y), \epsilon(y))$, where ϵ is defined as the concatenation

$$\begin{tikzcd} fg(y) \arrow[r, equal, "\mathsf{ap}_{fg}(G(y))^{-1}"] & fgfg(y) \arrow[r, equal, "\mathsf{ap}_f(H(g(y)))"] & fg(y) \arrow[r, equal, "G(y)"] & y. \end{tikzcd}$$

Now it remains to construct the contraction, which we do by Σ-induction. Let $x : A$, and let $p : f(x) = y$. Since $p : f(x) = y$ has a free endpoint, we can apply path induction on it. Our goal is now to construct an identification

$$(g(f(x)), \varepsilon(f(x))) = (x, \mathsf{refl}_{f(x)}).$$

We will construct an identification of the form $\mathsf{eq\_pair}(H(x), \_)$, so it remains to construct an identification of type

$$\mathsf{tr}_{f(-)=f(x)}(H(x), \varepsilon(f(x))) = \mathsf{refl}_{f(x)}.$$

Using Exercise 4.4 we see that

$$\mathsf{tr}_{f(-)=f(x)}(H(x), \varepsilon(f(x))) = \mathsf{ap}_f\,(H(x))^{-1} \cdot \epsilon(f(x)),$$

so it suffices to show that the square

$$\begin{tikzcd} fgfgf(x) \arrow[r, equal, "\mathsf{ap}_{fg}(G(f(x)))"] \arrow[d, equal, "\mathsf{ap}_f(H(gf(x)))"'] & fgf(x) \arrow[d, equal, "\mathsf{ap}_f(H(x))"] \\ fgf(x) \arrow[r, equal, "G(f(x))"'] & f(x) \end{tikzcd}$$

commutes, i.e. that

$$\mathsf{ap}_{fg}\,(G(f(x))) \cdot \mathsf{ap}_f\,(H(x)) = \mathsf{ap}_f\,(H(gf(x))) \cdot G(f(x)).$$

Recall from Definition 6.3.2 that we have $H(gf(x)) = \mathsf{ap}_{gf}\,(H(x))$ and $\mathsf{ap}_{fg}\,(G(y)) = G(fg(y))$. Using these two identifications and the fact that for any $p, p' : x = y$, $r : y = z$, $q, q' : x = y'$, and $s : y' = z$, if $p = p'$ and $q = q'$ then $p' \cdot r = q' \cdot s \to p \cdot r = q \cdot s$, 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. However, this is just a naturality square the homotopy $Gf : fgf \sim f$, which commutes by Definition 6.3.1. □

Corollary 6.3.4. *Let A be a type, and let $a : A$. Then the type*

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

is contractible.

Proof. By Theorem 5.2.5, the identity function is an equivalence. Therefore, the fibers of the identity function are contractible by Theorem 6.3.3. Note that $\sum_{(x:A)} x = a$ is exactly the fiber of id_A at $a : A$. □

Exercises

6.1 Show that if A is contractible, then for any $x, y : A$ the identity type $x = y$ is also contractible.

6.2 Suppose that A is a retract of B. Show that

$$\mathsf{is\_contr}(B) \to \mathsf{is\_contr}(A).$$

6.3 (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 5.5 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.

6.4 Let C be a contractible type with center of contraction $c : C$. Furthermore, let B be a type family over C. Show that the map $b \mapsto (c, b) : B(c) \to \sum_{(x:C)} B(x)$ is an equivalence.

6.5 Consider a type A with base point $a : A$, and let B be a type family on A that implies the identity type, i.e. there is a term

$$\alpha : \textstyle\prod_{(x:A)} B(x) \to (a = x).$$

Construct an equivalence

$$\Big(\textstyle\sum_{(x:A)} B(x)\Big) \simeq \Big(\textstyle\sum_{(y:B(a))} \alpha(a, y) = \mathsf{refl}_a\Big).$$

6.6 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.

6.7 Use Exercise 6.2 to show that if $A \times B$ is contractible, then A and B are contractible.

Lecture 7

The fundamental theorem of identity types

7.1 Fiberwise equivalences

Consider a family

$$f : \prod_{(x:A)} B(x) \to C(x)$$

of maps. Such f is also called a **fiberwise map** or **fiberwise transformation**.

Definition 7.1.1. We define a map

$$\mathsf{total}(f) : \sum_{(x:A)} B(x) \to \sum_{(x:A)} C(x).$$

by $\lambda(x,y).\,(x, f(x,y))$.

Lemma 7.1.2. *For any fiberwise transformation $f : \prod_{(x:A)} B(x) \to C(x)$, and any $a : A$ and $c : C(a)$, there is an equivalence*

$$\mathsf{fib}_{f(a)}(c) \simeq \mathsf{fib}_{\mathsf{total}(f)}((a,c)).$$

Proof. We first define a map $\varphi_{(a,c)} : \mathsf{fib}_{f(a)}(c) \to \mathsf{fib}_{\mathsf{total}(f)}((a,c))$ by Σ-induction. It suffices to define a term of type

$$\prod_{(b:B(a))} (f(a,b) = c) \to \mathsf{fib}_{\mathsf{total}(f)}((a,c))$$

for every $a : A$ and $c : C(a)$. Since the endpoint of the identification $f(a,b) = c$ is free, we may proceed by path induction. It suffices to construct a term of type $\prod_{(b:B(a))} \mathsf{fib}_{\mathsf{total}(f)}((a, f(a,b)))$. For $b : B$, we take

$$((a,b), \mathsf{refl}_{(a,f(a,b))}) : \mathsf{fib}_{\mathsf{total}(f)}((a, f(a,b))).$$

Since we have defined $\varphi_{(a,c)}$ for all $a : A$ and $c : C(a)$, note that we also obtain a map

$$\varphi_t : \mathsf{fib}_{f(\mathsf{pr}_1(t))}(\mathsf{pr}_2(t)) \to \mathsf{fib}_{\mathsf{total}(f)}(t)$$

for every $t : \sum_{(x:A)} C(x)$.

Our next goal is to define a map

$$\psi_{(a,c)} : \mathsf{fib}_{\mathsf{total}(f)}(a,c) \to \mathsf{fib}_{f(a)}(c).$$

for every $a : A$ and $c : C(a)$. In fact it will be easier to construct more generally the map

$$\psi_t : \mathsf{fib}_{\mathsf{total}(f)}(t) \to \mathsf{fib}_{f(\mathsf{pr}_1(t))}(\mathsf{pr}_2(t)).$$

for every $t : \sum_{(x:A)} C(x)$. Let us write $t_1 :\equiv \mathsf{pr}_1(t)$ and $t_2 :\equiv \mathsf{pr}_2(t)$. By Σ-induction it suffices to construct a term of type

$$\prod_{(s:\sum_{(x:A)} B(x))}(\mathsf{total}(f)(s) = t) \to \mathsf{fib}_{f(t_1)}(t_2),$$

We apply Σ-induction once more, and we need to construct a term of type

$$\prod_{(x:A)}\prod_{(y:B(x))}(\mathsf{total}(f)(x,y) = t) \to \mathsf{fib}_{f(t_1)}(t_2).$$

Let $x : A$ and $y : B(x)$. Since the endpoint of the identification $\mathsf{total}(f)(x,y) = t$ is free, we proceed by path induction. Our goal is now to construct a term of type

$$\mathsf{fib}_{f(x)}(f(x,y)).$$

Here we simply take $(y, \mathsf{refl}_{f(x,y)})$, completing the construction of ψ.

To show that ψ is a retraction of φ, we construct an identification

$$\psi(\varphi((b,p))) = (b,p)$$

for each $b : B(a)$ and $p : f(a,b) = c$. We proceed by path induction on $p : f(a,b) = c$. Our goal is now to show that

$$\psi(\varphi(b, \mathsf{refl}_{f(a,b)})) = (b, \mathsf{refl}_{f(a,b)})$$

By definition we have

$$\varphi(b, \mathsf{refl}_{f(a,b)}) \equiv ((a,b), \mathsf{refl}_{(a,f(a,b))})$$
$$\psi((a,b), \mathsf{refl}_{(a,f(a,b))}) \equiv (b, \mathsf{refl}_{f(a,b)}).$$

Thus, we simply take $\mathsf{refl}_{(b,\mathsf{refl}_{f(a,b)})}$ to complete the goal of showing that ψ is a retraction of φ.

To show that ψ_t is a section of φ_t for every $t : \sum_{(x:A)} C(x)$, we construct an identification

$$\varphi(\psi((x, b), r)) = ((x, b), r)$$

for each $x : A$, $y : B(x)$, and $r : \mathsf{total}(f)(x, y) = t$. We proceed by path induction on r, so our goal is to show

$$\varphi(\psi((x, y), \mathsf{refl}_{(x,f(x,y))})) = (x, f(x, y)).$$

By definition we have

$$\begin{aligned} \psi((x, y), \mathsf{refl}_{(x,f(x,y))}) &\equiv (y, \mathsf{refl}_{f(x,y)}) \\ \varphi(y, \mathsf{refl}_{f(x,y)}) &\equiv ((x, y), \mathsf{refl}_{(x,f(x,y))}) \end{aligned}$$

Thus, we simply take $\mathsf{refl}_{(x,f(x,y))}$ to complete the goal of showing that ψ is a section of φ. □

Theorem 7.1.3. *Let $f : \prod_{(x:A)} B(x) \to C(x)$ be a fiberwise transformation. The following are logically equivalent:*

*(i) For each $x : A$, the map $f(x)$ is an equivalence. In this case we say that f is a **fiberwise equivalence**.*

(ii) The map $\mathsf{total}(f) : \sum_{(x:A)} B(x) \to \sum_{(x:A)} C(x)$ is an equivalence.

Proof. By Theorems 6.2.3 and 6.3.3 it suffices to show that $f(x)$ is a contractible map for each $x : A$, if and only if $\mathsf{total}(f)$ is a contractible map. Thus, we will show that $\mathsf{fib}_{f(x)}(c)$ is contractible if and only if $\mathsf{fib}_{\mathsf{total}(f)}(x, c)$ is contractible, for each $x : A$ and $c : C(x)$. However, by Lemma 7.1.2 these types are equivalent, so the result follows by Exercise 6.3. □

7.2 The fundamental theorem

Theorem 7.2.1. *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:*

(i) The canonical family of maps

$$\mathsf{rec}_{a=}(b) : \prod_{(x:A)}(a = x) \to B(x)$$

is a fiberwise equivalence.

(ii) The total space

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

is contractible.

Proof. By Theorem 7.1.3 it follows that the fiberwise transformation $\mathsf{rec}_{a=}(b)$ is a fiberwise equivalence if and only if it induces an equivalence

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

on total spaces. We have that $\sum_{(x:A)} a = x$ is contractible. Now it follows by Exercise 6.3, applied in the case

$$\begin{tikzcd} \sum_{(x:A)} a = x \arrow[rr, "\mathsf{total}(\mathsf{rec}_{a=}(b))"] \arrow[dr, "\simeq"'] & & \sum_{(x:A)} B(x) \arrow[dl] \\ & \mathbf{1} & \end{tikzcd}$$

that $\mathsf{total}(\mathsf{rec}_{a=}(b))$ is an equivalence if and only if $\sum_{(x:A)} B(x)$ is contractible. □

As an application of the fundamental theorem we show that equivalences are embeddings. The notion embeddings is the homotopically correct notion of injective maps.

Definition 7.2.2. 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 $\mathsf{is\_emb}(f)$ for the type of witnesses that f is an embedding.

Theorem 7.2.3. *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 7.2.1 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 7.1.3, since for each $y : A$ the map

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

is an equivalence by Exercise 5.3. The fiber $\mathsf{fib}_e(e(x))$ is contractible by Theorem 6.3.3, so it follows by Exercise 6.3 that the type $\sum_{(y:A)} e(x) = e(y)$ is indeed contractible. □

Exercises

7.1 (a) Let $f, g : \prod_{(x:A)} B(x) \to C(x)$ be two fiberwise transformations. Show that

$$\left(\textstyle\prod_{(x:A)} f(x) \sim g(x)\right) \to (\mathsf{total}(f) \sim \mathsf{total}(g)).$$

(b) Let $f : \prod_{(x:A)} B(x) \to C(x)$ and let $g : \prod_{(x:A)} C(x) \to D(x)$. Show that

$$\mathsf{total}(\lambda x.\, g(x) \circ f(x)) \sim \mathsf{total}(g) \circ \mathsf{total}(f).$$

(c) For any family B over A, show that

$$\mathsf{total}(\lambda x.\, \mathsf{id}_{B(x)}) \sim \mathsf{id}_{\sum_{(x:A)} B(x)}.$$

7.2 Construct an equivalence

$$\left(\textstyle\sum_{(x:A)} f(x) = y\right) \simeq \left(\textstyle\sum_{(x:A)} y = f(x)\right).$$

7.3 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 fiberwise transformation

$$\mathsf{fib\_triangle}(h, H) : \textstyle\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(x).$$

(b) Show that h is an equivalence if and only if $\mathsf{fib\_triangle}(h, H)$ is a fiberwise equivalence.

7.4 Let $f : A \to B$ be a map. Then for each $y : B$, and each $(x, p), (x', p') : \mathsf{fib}_f(y)$, the map

$$(x, p) = (x', p') \to \textstyle\sum_{(q:x=x')} p = \mathsf{ap}_f\,(q) \bullet p'$$

is an equivalence.

7.5 Let B be a type family over A. Show that for the projection map $\mathsf{pr}_1 : (\sum_{(x:A)} B(x)) \to A$, the map

$$\lambda y.\,((x, y), \mathsf{refl}_x) : B(x) \to \mathsf{fib}_{\mathsf{pr}_1}(x)$$

is an equivalence, for each $x : A$. Conclude that pr_1 is itself an equivalence if and only if each $B(x)$ is contractible.

7.6 Let A be a type with $a : A$, and let B be a type family over A. Show that for *any* family of maps

$$f : \textstyle\prod_{(x:A)} (a = x) \to B(x)$$

the following are logically equivalent:

(i) The family of maps f is a fiberwise equivalence.

(ii) The total space

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

is contractible.

7.7 Let $a : A$, and let B be a type family over A. Use Exercises 7.1, 6.2 and 7.6 to show that if each $B(x)$ is a retract of $a = x$, then $B(x)$ is equivalent to $a = x$ for every $x : A$.

7.8 Show that the map $\mathbf{0} \to A$ is an embedding for every type A.

7.9 Show that

$$(f \sim g) \to (\mathsf{is\_emb}(f) \leftrightarrow \mathsf{is\_emb}(g))$$

for any $f, g : A \to B$.

7.10 Consider a commuting triangle

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

with $H : f \sim g \circ e$, where e is an equivalence. Show that f is an embedding if and only if g is an embedding.

7.11 Consider a map

$$f : A \to \sum_{(y:B)} C(y).$$

(a) Construct a fiberwise transformation

$$f' : \prod_{(y:B)} \mathsf{fib}_{\mathsf{pr}_1 \circ f}(y) \to C(y).$$

(b) Construct an equivalence

$$\mathsf{fib}_{f'(b)}(c) \simeq \mathsf{fib}_f((b,c))$$

for every $(b,c) : \sum_{(y:B)} C(y)$.

(c) Conclude that the following are equivalent:

(i) f is an equivalence.

(ii) f' is a fiberwise equivalence.

Lecture 8

The hierarchy of homotopical complexity

8.1 Propositions and subtypes

Definition 8.1.1. A type A is said to be a **proposition** if there is a term of type

$$\mathsf{is\_prop}(A) :\equiv \prod_{(x,y:A)} \mathsf{is\_contr}(x = y).$$

Example 8.1.2. Any contractible type is a proposition by Exercise 6.1. However, propositions do not need to be inhabited: the empty type is also a proposition, since

$$\prod_{(x,y:\mathbf{0})} \mathsf{is\_contr}(x = y)$$

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**.

Lemma 8.1.3. *Let A be a type. Then we have*

$$\mathsf{is\_prop}(A) \leftrightarrow \Big(\prod_{(x,y:A)} x = y\Big).$$

Proof. Suppose A is a proposition. By taking the center of contraction of $x = y$ for each $x, y : A$ we obtain a term of type $\prod_{(x,y:A)} x = y$.

Now suppose that A is a type equipped with $H : \prod_{(x,y:A)} x = y$. Then we take $H(x,x)^{-1} \cdot H(x,y)$ as the center of contraction of $x = y$. To construct the contraction

$$\prod_{(p:x=y)} H(x,x)^{-1} \cdot H(x,y) = p$$

we proceed by path induction. Our goal is to show that

$$H(x,x)^{-1} \cdot H(x,x) = \mathsf{refl}_x. \qquad \square$$

By proof irrelevance it follows that propositions are contractible as soon as they are inhabited.

Corollary 8.1.4. *For any type A we have*

$$\mathsf{is\_prop}(A) \leftrightarrow (A \to \mathsf{is\_contr}(A)).$$

Lemma 8.1.5. *Let A and B be types, and let $e : A \simeq B$. Then we have*

$$\mathsf{is\_prop}(A) \leftrightarrow \mathsf{is\_prop}(B).$$

Proof. We will show that $\mathsf{is\_prop}(B)$ implies $\mathsf{is\_prop}(A)$. 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 7.2.3 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 6.3.3. □

In type theory terms always come equipped with their types, i.e. they never appear in isolation. This is useful from the perspective that terms are programs with a certain specification, but as a consequence we cannot consider subtypes in the same way as set theorists have subsets. Our definition of subtype is therefore considerably different:

Definition 8.1.6. 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.

We will show in Corollary 8.3.9 that a type family B over A is a subtype of A if and only if the projection map $\mathsf{pr}_1 : \big(\sum_{(x:A)} B(x)\big) \to A$ is an embedding.

8.2 Sets

Definition 8.2.1. A type A is said to be a **set** if there is a term of type

$$\mathsf{is\_set}(A) :\equiv \textstyle\prod_{(x,y:A)} \mathsf{is\_prop}(x = y).$$

Lemma 8.2.2. *A type A is a set if and only if it satisfies* ***axiom K****, which asserts that*

$$\prod_{(x:A)}\prod_{(p:x=x)}\mathsf{refl}_x = p.$$

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. □

Lemma 8.2.3. *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 $\rho : \prod_{(x:A)} R(x, x)$.

Then any fiberwise map

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

is a fiberwise equivalence. Consequently, if there is such a fiberwise map, then 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 fiberwise transformation

$$\mathsf{rec}_{x=}(\rho(x)) : \prod_{(y:A)}(x = y) \to R(x, y).$$

Since each $R(x, y)$ is assumed to be a proposition, it therefore follows that each $R(x, y)$ is a retract of $x = y$. We conclude by Exercise 7.7 that for each $x, y : A$, the map $f(x, y) : R(x, y) \to (x = y)$ must be an equivalence.

Now it also follows that A is a set, since its identity types are equivalent to propositions, and therefore they are propositions by Lemma 8.1.5. □

Theorem 8.2.4. *The type of natural numbers is a set.*

Proof. We will apply Lemma 8.2.3. Note that the observational equality $\mathrm{Eq}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathcal{U})$ on $\mathbb{N}$ (Definition 3.2.6) is a reflexive relation by Exercise 3.3, and moreover that $\mathrm{Eq}_{\mathbb{N}}(n, m)$ is a proposition for every $n, m : \mathbb{N}$ (proof by double induction). Therefore it suffices to show that

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

This follows from the fact that observational equality is the *least* reflexive relation, which was shown in Exercise 3.4. □

8.3 General truncation levels

Definition 8.3.1. We define $\mathsf{is\_trunc} : \mathbb{Z}_{\geq -2} \to \mathcal{U} \to \mathcal{U}$ by induction on $k : \mathbb{Z}_{\geq -2}$, taking

$$\begin{aligned} \mathsf{is\_trunc}_{-2}(A) &:\equiv \mathsf{is\_contr}(A) \\ \mathsf{is\_trunc}_{k+1}(A) &:\equiv \textstyle\prod_{(x,y:A)} \mathsf{is\_trunc}_k(x = y). \end{aligned}$$

For any type A, we say that A is **k-truncated**, or a **k-type**, if there is a term of type $\mathsf{is\_trunc}_k(A)$. We say that a map $f : A \to B$ is k-truncated if its fibers are k-truncated.

Theorem 8.3.2. *If A is a k-type, then A is also a $(k+1)$-type.*

Proof. We have seen in Example 8.1.2 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 8.3.3. *If $e : A \simeq B$ is an equivalence, and B is a k-type, then so is A.*

Proof. We have seen in Exercise 6.3 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 7.2.3 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 8.3.4. *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 8.3.3 that $x = y$ is a k-type for every $x, y : A$. In other words, A is a $(k+1)$-type. □

In the following definition we generalize the notion of contractible map.

Definition 8.3.5. We say that a map $f : A \to B$ is **k-truncated** if for each $y : B$ the fiber $\mathsf{fib}_f(y)$ is k-truncated.

Theorem 8.3.6. *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.

(ii) The projection map

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

is k-truncated.

Proof. By Exercises 7.3 and 6.6 we obtain equivalences

$$B(x) \simeq \mathsf{fib}_{\mathsf{pr}_1}(x)$$

for every $x : A$. Therefore the claim follows from Theorem 8.3.3. □

Theorem 8.3.7. *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 using Exercise 4.4 and basic manipulations of identifications that

$$\begin{aligned}((x,p) = (y,q)) &\simeq \textstyle\sum_{(r:x=y)} \mathsf{tr}_{f(-)=b}(r,p) = q \\ &\simeq \textstyle\sum_{(r:x=y)} \mathsf{ap}_f\,(r)^{-1} \cdot p = q \\ &\simeq \textstyle\sum_{(r:x=y)} \mathsf{ap}_f\,(r) = p \cdot q^{-1} \\ &\equiv \mathsf{fib}_{\mathsf{ap}_f}(p \cdot q^{-1}).\end{aligned}$$

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 8.3.3 that f is $(k+1)$-truncated.

For the converse, note that we have equivalences

$$\mathsf{fib}_{\mathsf{ap}_f}(p) \simeq ((x,p) = (y, \mathsf{refl}_{f(y)})).$$

Therefore 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 $p : f(x) = f(y)$, and therefore $\mathsf{fib}_{\mathsf{ap}_f}(p)$ is k-truncated by Theorem 8.3.3. □

Corollary 8.3.8. *A map is an embedding if and only if its fibers are propositions.*

Corollary 8.3.9. *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 8.3.10. *Let $f : \prod_{(x:A)} B(x) \to C(x)$ be a fiberwise transformation. Then the following are equivalent:*

(i) For each $x : A$ the map $f(x)$ is k-truncated.

(ii) The induced map

$$\mathsf{total}(f) : \left(\textstyle\sum_{(x:A)} B(x)\right) \to \left(\textstyle\sum_{(x:A)} C(x)\right)$$

is k-truncated.

Proof. This follows directly from Lemma 7.1.2 and Theorem 8.3.3. □

Exercises

8.1 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

$$\mathsf{is\_equiv}(\delta_A) \leftrightarrow \mathsf{is\_prop}(A).$$

(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.

8.2 (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 $\sum_{(x:A)} B(x)$. Hint: for the base case, use Exercises 6.3 and 6.4.

(b) Show that if A and B are k-types, then so is $A \times B$.

8.3 Show that $\mathbf{2}$ is a set by applying Lemma 8.2.3 with the observational equality on $\mathbf{2}$ defined in Exercise 3.8.

8.4 Show that for any two sets A and B, the disjoint sum $A + B$ is again a set.

8.5 (Hedberg's theorem) A type A is said to have **decidable equality** if there is a term of type

$$\prod_{(x,y:A)}(x = y) + \neg(x = y).$$

For any type A, and every $x, y : A$, consider the type family $D(x, y) : ((x = y) + \neg(x = y)) \to \mathcal{U}$ given by

$$D(x, y, \mathsf{inl}(p)) :\equiv \mathbf{1}$$
$$D(x, y, \mathsf{inr}(p)) :\equiv \mathbf{0}.$$

Use D to show that any type with decidable equality is a set.

8.6 Show that $\mathbb{N}$ and $\mathbf{2}$ have decidable equality, as defined in Exercise 8.5.

8.7 Show that if A and B have decidable equality, then so do $A + B$ and $A \times B$.

8.8 Use Exercises 6.2 and 5.9 to show that if A is a retract of a k-type B, then A is also a k-type.

8.9 A map $f : A \to B$ between sets is said to be **injective** if for every $x, y : A$ there is a map

$$(f(x) \to f(y)) \to (x = y).$$

(a) Use Exercise 8.3 to show that $\mathsf{const}_{0_\mathbf{2}}, \mathsf{const}_{1_\mathbf{2}} : \mathbf{1} \to \mathbf{2}$ are injective maps.

(b) Show that between sets are injective if and only if they are embeddings.

(c) Show that a type A is a set if and only if the map $\mathsf{const}_x : \mathbf{1} \to A$ is an embedding for every $x : A$.

Lecture 9

Function extensionality

A significant part of the development of homotopy type theory involves answering the following basic questions:

(i) What is the identity type of a given type A?

(ii) What is the total space of a type family B over A?

(iii) What are the fibers of a given map $f : A \to B$?

(iv) What does transporting a point in a given type family B over A do?

We have already characterized the identity types of Σ-types as a Σ-type of identity types (Theorem 5.3.1), of **0** and **1** since both are propositions (Example 8.1.2), of natural numbers $\mathbb{N}$ as observational equality (Theorem 8.2.4), and also of **2** as observational equality (Exercise 8.3). In this section we will discuss the identity type of Π-types.

9.1 Equivalent forms of function extensionality

Theorem 9.1.1. *e following are equivalent:*

(i) The principle of ***homotopy induction****: for every $f : \prod_{(x:A)} B(x)$ and every type family*

$$\Gamma, g : \textstyle\prod_{(x:A)} B(x), H : f \sim g \vdash P(g, H) \text{ type},$$

the map

$$\Big(\textstyle\prod_{(g:\prod_{(x:A)} B(x))} \prod_{(H:f\sim g)} P(g, H)\Big) \to P(f, \mathsf{htpy.refl}_f)$$

given by $s \mapsto s(f, \mathsf{htpy.refl}_f)$ has a section.

(ii) *The **function extensionality principle**: For every type family B over A, and any two dependent functions $f, g : \prod_{(x:A)} B(x)$, the canonical map*

$$\mathsf{htpy\_eq}(f,g) : (f = g) \to (f \sim g)$$

by path induction (sending refl_f to $\lambda x.\, \mathsf{refl}_{f(x)}$) is an equivalence. We will write $\mathsf{eq\_htpy}$ for its inverse.

(iii) *The **weak function extensionality principle** holds: For every type family B over A one has*

$$\Big(\prod_{(x:A)} \mathsf{is\_contr}(B(x))\Big) \to \mathsf{is\_contr}\Big(\prod_{(x:A)} B(x)\Big).$$

Proof. To show that homotopy induction implies function extensionality, note that from homotopy induction and Σ-induction combined, we obtain that for any type family P over $\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$, the evaluation map

$$\Big(\prod_{(t:\sum_{(g:\prod_{(x:A)} B(x))} f \sim g)} P(t)\Big) \to P((f, \mathsf{htpy.refl}_f))$$

has a section. In other words, the type $\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$ satisfies singleton induction and therefore it is contractible. Now we conclude by Theorem 7.2.1 that the canonical map $(f = g) \to (f \sim g)$ is an equivalence, i.e. that function extensionality holds.

Conversely, to prove homotopy induction from function extensionality we again note that by function extensionality and Theorem 7.2.1 the type

$$\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$$

is contractible. Therefore it satisfies singleton induction, so homotopy induction follows.

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.

To prove function extensionality from weak function extensionality, observe that it suffices by Theorem 7.2.1 to show that

$$\textstyle\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$$

is contractible.

Since the type $\sum_{(b:B(x))} f(x) = b$ is contractible for each $x : X$, it follows by our assumption of weak function extensionality that the type $\prod_{(x:A)} \sum_{(b:B(x))} f(x) = b$ is contractible. By Exercise 6.2 it therefore suffices to show that

$$\textstyle\sum_{(g:\prod_{(x:A)} B(x))} f \sim g$$

is a retract of the type $\prod_{(x:A)} \sum_{(b:B(x))} f(x) = b$. We have the functions

$$\begin{aligned} i &:\equiv \mathsf{ind}_\Sigma(\lambda g.\, \lambda H.\, \lambda x.\, (g(x), H(x))) \\ r &:\equiv \lambda p.\, (\lambda x.\, \mathsf{pr}_1(p(x)), \lambda x.\, \mathsf{pr}_2(p(x))). \end{aligned}$$

It remains to show that $r \circ i \sim \mathsf{id}$. This homotopy is constructed by Σ-induction. Let $g : \prod_{(x:A)} B(x)$ and let $H : f \sim g$. Then we have

$$\begin{aligned} r(i(g,H)) &\equiv r(\lambda x.\, (g(x), H(x))) \\ &\equiv (\lambda x.\, g(x), \lambda x.\, H(x)) \\ &\equiv (g, H). \end{aligned}$$

In other words, the homotopy $r \circ i \sim \mathsf{id}$ is given by $\mathsf{ind}_\Sigma(\lambda g.\, \lambda H.\, \mathsf{refl}_{((g,H))})$. □

Theorem 9.1.2. *Assume function extensionality. Then for any type family B over A one has*

$$\Big(\textstyle\prod_{(x:A)} \mathsf{is\_trunc}_k(B(x))\Big) \to \mathsf{is\_trunc}_k\Big(\textstyle\prod_{(x:A)} B(x)\Big).$$

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 9.1.1.

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 8.3.3 that $f = g$ is an k-type, and hence that $\prod_{(x:A)} B(x)$ is an $(k+1)$-type. □

Corollary 9.1.3. *Suppose B is a k-type. Then $A \to B$ is also a k-type, for any type A.*

The following theorem is sometimes called the **type theoretic principle of choice**. This terminology comes from the point of view of propositions as types, where the Σ-type has the role of the existential quantifier, and the Π-type has the role of the universal quantifier.

Theorem 9.1.4. *Let $C(x,y)$ be a type in context $\Gamma, x : A, y : B(x)$. Then the map*

$$\varphi : \Big(\prod_{(x:A)}\sum_{(y:B(x))}C(x,y)\Big) \to \Big(\sum_{(f:\prod_{(x:A)} B(x))}\prod_{(x:A)}C(x,f(x))\Big)$$

given by $\lambda h.\, (\lambda x.\, \mathsf{pr}_1(h(x)), \lambda x.\, \mathsf{pr}_2(h(x)))$ is an equivalence.

Proof. The map ψ in the converse direction is defined by

$$\psi :\equiv \mathsf{ind}_\Sigma(\lambda f.\, \lambda g.\, \lambda x.\, (f(x), g(x))).$$

We need to define homotopies $\psi \circ \varphi \sim \mathsf{id}$ and $\varphi \circ \psi \sim \mathsf{id}$.

For the first homotopy, let $h : \prod_{(x:A)} \sum_{(y:B(x))} C(x,y)$. Then we have

$$\begin{aligned}\psi(\varphi(h)) &\equiv \psi(\lambda x.\, \mathsf{pr}_1(h(x)), \lambda x.\, \mathsf{pr}_2(h(x))) \\ &\equiv \lambda x.\, (\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x))).\end{aligned}$$

Note that for each $x : A$ we have an identification

$$(\mathsf{pr}_1(h(x)), \mathsf{pr}_2(h(x))) = h(x).$$

Therefore we obtain a *homotopy* $\psi(\varphi(h)) \sim h$, but this suffices because function extensionality now provides us with an identification $\psi(\varphi(h)) = h$.

The second homotopy is constructed by Σ-induction. Let $f : \prod_{(x:A)} B(x)$ and let $g : \prod_{(x:A)} C(x, f(x))$. Then we have

$$\begin{aligned}\varphi(\psi(f,g)) &\equiv \varphi(\lambda x.\, (f(x), g(x))) \\ &\equiv (\lambda x.\, f(x), \lambda x.\, g(x)) \\ &\equiv (f,g)\end{aligned}$$

where the last judgmental equality holds by the η-rule for Π-types. In other words, the homotopy $\varphi \circ \psi \sim \mathsf{id}$ is given by

$$\mathsf{ind}_\Sigma(\lambda f.\, \lambda g.\, \mathsf{refl}_{(f,g)}).$$ □

Corollary 9.1.5. *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.

9.2 Universal properties

The function extensionality principle allows us to prove *universal properties*: characterizations of all maps out of (or into) a given type. Universal properties characterize a type up to equivalence. In the following theorem we prove the universal property of dependent pair types.

Theorem 9.2.1. *Let B be a type family over A, and let X be a type. Then the map*

$$\mathsf{ev\_pair} : \Big(\Big(\textstyle\sum_{(x:A)} B(x)\Big) \to X\Big) \to \Big(\textstyle\prod_{(x:A)} (B(x) \to X)\Big)$$

given by $f \mapsto \lambda a.\, \lambda b.\, f((a,b))$ is an equivalence.

This theorem justifies that we usually write $f(a,b)$ rather than $f((a,b))$, since $f : \big(\sum_{(x:A)} B(x)\big) \to X$ is uniquely determined by its action on terms of the form (a,b).

Proof. The map in the converse direction is simply

$$\mathsf{ind}_\Sigma : \Big(\textstyle\prod_{(x:A)} (B(x) \to X)\Big) \to \Big(\Big(\textstyle\sum_{(x:A)} B(x)\Big) \to X\Big).$$

By the computation rules for Σ-types we have

$$\lambda f.\, \mathsf{refl}_f : \mathsf{ev\_pair} \circ \mathsf{ind}_\Sigma \sim \mathsf{id}$$

To show that $\mathsf{ind}_\Sigma \circ \mathsf{ev\_pair} \sim \mathsf{id}$ 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

$$\textstyle\prod_{(t:\sum_{(x:A)} B(x))} \mathsf{ind}_\Sigma\big(\lambda x.\, \lambda y.\, f((x,y))\big)(t) = f(t).$$

We obtain this homotopy by another application of Σ-induction. □

Corollary 9.2.2. *Let A, B, and X be types. Then the map*

$$\mathsf{ev\_pair} : (A \times B \to X) \to (A \to (B \to X))$$

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 9.2.3. *Let B be a type family over A, and let $a : A$. Then the map*

$$\mathsf{ev\_refl} : \Big(\textstyle\prod_{(x:A)}(a = x) \to B(x)\Big) \to B(a)$$

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 $\mathsf{ev\_refl} \circ \varphi \sim \mathsf{id}$.

To see that $\varphi \circ \mathsf{ev\_refl} \sim \mathsf{id}$, 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

$$\textstyle\prod_{(x:A)}\prod_{(p:a=x)}\varphi(f(a, \mathsf{refl}_a), x, p) = f(x, p).$$

This follows by path induction on p, since $\varphi(f(a, \mathsf{refl}_a), a, \mathsf{refl}_a) \equiv f(a, \mathsf{refl}_a)$. □

9.3 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 9.3.3 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

$$- \circ f : \Big(\textstyle\prod_{(y:B)}P(y)\Big) \to \Big(\textstyle\prod_{(x:A)}P(f(x))\Big)$$

is an equivalence. However, in our proof of this fact we will rely on a piece of data that every equivalence satisfies in addition to being invertible.

Definition 9.3.1. We say that a map $f : A \to B$ is a **half-adjoint equivalence**, in the sense that there are

$$\begin{aligned} g &: B \to A \\ G &: f \circ g \sim \mathsf{id}_B \\ H &: g \circ f \sim \mathsf{id}_A \\ K &: G \cdot f \sim f \cdot H. \end{aligned}$$

We write $\mathsf{half\_adj}(f)$ for the type of such quadruples (g, G, H, K).

To show that every equivalence is a half-adjoint equivalence, we also introduce the notion of *path-split* maps.

Definition 9.3.2. 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 $\mathsf{path\_split}(f)$ for the type

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

Theorem 9.3.3. *For any map $f : A \to B$, the following are equivalent:*

(i) f is an equivalence.

(ii) f is path-split.

(iii) f is a half-adjoint equivalence.

(iv) 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 $s \mapsto s \circ f$ is an equivalence.

(v) 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 see that (i) implies (ii) we note that any equivalence has a section, and its action on paths is an equivalence by Theorem 7.2.3 so again it has a section.

To show that (ii) implies (iii), assume that f is path-split. Thus we have $(g, G) : \mathsf{sec}(f)$, and the assumption that $\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$ has a section for every $x, y : A$ gives us a term of type

$$\prod_{(x:A)} \mathsf{fib}_{\mathsf{ap}_f}(G(f(x))).$$

By Theorem 9.1.4 this type is equivalent to

$$\sum_{(H:\prod_{(x:A)} g(f(x))=x)} \prod_{(x:A)} G(f(x)) = \mathsf{ap}_f\,(H(x))\,,$$

so we obtain $H : g \circ f \sim \mathsf{id}_A$ and $K : G \cdot f \sim f \cdot H$, showing that f is a half-adjoint equivalence.

To show that (iii) implies (iv), suppose that f comes equipped with (g, G, H, K) witnessing that f is a half-adjoint equivalence. 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 $s \mapsto \lambda y.\, \mathsf{tr}_P(G(y), sg(y))$.

To see that φ is a section of $- \circ f$, let $s : \prod_{(x:A)} P(f(x))$. By function extensionality it suffices to construct a homotopy $\varphi(s) \circ f \sim s$. In other words, we have to show that

$$\mathsf{tr}_P(G(f(x)), s(g(f(x))) = s(x)$$

for any $x : A$. Now we use the additional homotopy K from our assumption that f is a half-adjoint equivalence. Since we have $K(x) : G(f(x)) = \mathsf{ap}_f\,(H(x))$ it suffices to show that

$$\mathsf{tr}_P(\mathsf{ap}_f\,(H(x))\,, sgf(x)) = s(x).$$

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

$$\mathsf{tr}_{P \circ f}(H(x), sgf(x)) = s(x).$$

We have such an identification by $\mathsf{apd}_{H(x)}(s)$.

To see that φ is a retraction of $- \circ f$, let $s : \prod_{(y:B)} P(y)$. By function extensionality it suffices to construct a homotopy $\varphi(s \circ f) \sim s$. In other words, we have to show that

$$\mathsf{tr}_P(G(y), sfg(y)) = s(y)$$

for any $y : B$. We have such an identification by $\mathsf{apd}_{G(y)}(s)$. This completes the proof that (iii) implies (iv).

Note that (v) is an immediate consequence of (iv), since we can just choose P to be the constant family X.

It remains to show that (v) 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 6.3.3. In particular, choosing $X \equiv A$ we see that the fiber

$$\mathsf{fib}_{-\circ f}(\mathsf{id}_A) \equiv \textstyle\sum_{(h:B\to A)} h \circ f = \mathsf{id}_A$$

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

9.1 (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.

9.2 (a) Show that for any type A the type $\mathsf{is\_contr}(A)$ is a proposition.

(b) Show that for any type A and any $k \geq -2$, the type $\mathsf{is\_trunc}_k(A)$ is a proposition.

9.3 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 $\mathsf{is\_equiv}(f)$ is a proposition.

(d) Use Exercises 9.1 and 9.2 to show that $\mathsf{is\_equiv}(f) \simeq \mathsf{is\_contr}(f)$.

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.

9.4 Show that $\mathsf{path\_split}(f)$ and $\mathsf{half\_adj}(f)$ are propositions for any map $f : A \to B$.

9.5 Let $f : X \to Y$ be a map. Show that the following are equivalent:

(i) f is an equivalence.

(ii) For any type A, the map $f \circ - : X^A \to Y^A$ is an equivalence.

9.6 Construct for any type A an equivalence

$$\mathsf{is\_invertible}(\mathsf{id}_A) \simeq \Big(\mathsf{id}_A \sim \mathsf{id}_A\Big).$$

9.7 (a) Show that the map

$$(\mathbf{0} \to X) \to \mathbf{1}$$

given by $\lambda f.\, \star$ is an equivalence.

(b) Show that any type Y satisfying the property that the map

$$\lambda f.\, \star : (Y \to X) \to \mathbf{1}$$

is an equivalence for any X, is itself equivalent to $\mathbf{0}$.

9.8 (a) Show that the map

$$(A + B \to X) \to \Big((A \to X) \times (B \to X)\Big)$$

given by $f \mapsto (f \circ \mathsf{inl}, f \circ \mathsf{inr})$ is an equivalence.

(b) Show that any type Y that is equipped with functions $i : A \to Y$ and $j : B \to Y$ and satisfies the condition that the map

$$\lambda f.\, (f \circ i, f \circ j) : (Y \to X) \to \Big((A \to X) \times (B \to X)\Big)$$

is an equivalence for any X, is itself equivalent to $A + B$.

9.9 (a) Show that the map

$$(\mathbf{1} \to X) \to X$$

given by $\lambda f.\, f(\star)$ is an equivalence.

(b) Show that a type C is contractible if and only if it comes equipped with a term $c : C$ and satisfies the condition that the map

$$\lambda f.\, f(c) : (C \to X) \to X$$

is an equivalence for any X.

9.10 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 $\mathsf{sec}(f)$.

9.11 Let $e_i : A_i \simeq B_i$ be an equivalence for every $i : I$. Show that the map

$$\lambda f.\, \lambda i.\, e_i \circ f : \Big(\prod_{(i:I)} A_i\Big) \to \Big(\prod_{(i:I)} B_i\Big)$$

is an equivalence.

9.12 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 fiberwise transformations

$$\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(x).$$

(b) Show that the type $\sum_{(h:A\simeq B)} f \sim g \circ h$ is equivalent to the type of fiberwise equivalences

$$\prod_{(x:X)} \mathsf{fib}_f(x) \simeq \mathsf{fib}_g(x).$$

9.13 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 fiberwise transformations

$$\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(h(x)).$$

9.14 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

$$\begin{aligned} f &: A \to B \\ g &: B \to A \\ H &: f \circ g = \mathsf{id}_B \\ K &: g \circ f = \mathsf{id}_A. \end{aligned}$$

Lecture 10

Homotopy pullbacks

Suppose we are given a map $f : A \to B$, and type families P over A, and Q over B. Then any fiberwise map

$$g : \prod_{(x:A)} P(x) \to Q(f(x))$$

gives rise to a commuting square

$$\begin{tikzcd}
\sum_{(x:A)} P(x) \arrow[r, "\mathsf{total}_f(g)"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(y:B)} Q(y) \arrow[d, "\mathsf{pr}_1"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

where $\mathsf{total}_f(g)$ is defined as $\lambda(x,p).\,(f(x), g(x,y))$. We will show in Theorem 10.5.2 that g is a fiberwise equivalence if and only if this square is a *pullback square*. This generalization of Theorem 7.1.3 is therefore abstracting away from the notion of fiberwise equivalence, and it serves as our motivating theorem to introduce pullbacks. The connection between pullbacks and fiberwise equivalences has an important role in the descent theorem in Lecture 14.

10.1 Cartesian squares

Recall that a square

$$\begin{tikzcd}
C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\
A \arrow[r, "g"'] & X
\end{tikzcd}$$

is said to **commute** if there is a homotopy $H : f \circ p \sim g \circ q$. The pullback property is a *universal property* of the upper left corner of a commuting square (in our case C), characterizing the maps *into* it.

To describe the universal property of pullbacks we first need to have a closer look at the *anatomy* of commuting squares.

Definition 10.1.1. A commuting square

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$ can be dissected into three parts, consisting of a *cospan*, a type, and a *cone*, where

(i) A **cospan** consists of three types A, X, and B, and maps $f : A \to X$ and $g : B \to X$.

(ii) Given a type C, a **cone** on the cospan $A \overset{f}{\to} X \overset{g}{\leftarrow} B$ with **vertex** C consists of maps $p : C \to A$, $q : C \to B$ and a homotopy $H : f \circ p \sim g \circ q$. 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.

Given a cone with vertex C on a span $A \overset{f}{\to} X \overset{g}{\leftarrow} B$ and a map $h : C' \to C$, we construct a new cone with vertex C' in the following definition.

Definition 10.1.2. For any cone (p, q, H) with vertex C and any type C', we define a map
$$\mathsf{cone\_map}(p, q, H) : (C' \to C) \to \mathsf{cone}(C')$$
by $h \mapsto (p \circ h, q \circ h, H \circ h)$.

Definition 10.1.3. 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

$$\mathsf{cone\_map}(p, q, H) : (C' \to C) \to \mathsf{cone}(C')$$

is an equivalence for every type C'.

We often indicate the universal property with a diagram as follows:

$$\begin{tikzcd}
C' \arrow[drr, bend left, "q'"] \arrow[dr, dotted, "h"] \arrow[ddr, bend right, "p'"'] & & \\
& C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\
& A \arrow[r, "f"'] & X
\end{tikzcd}$$

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 $\alpha : \mathsf{cone\_map}((p, q, H), h) = (p', q', H')$ is contractible by Theorem 6.3.3.

In order to see what goes on in the universal property of pullbacks, we need to first characterize the identity type of $\mathsf{cone}(C)$, for any type C.

Lemma 10.1.4. *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*

$$\begin{aligned} K &: p \sim p' \\ L &: q \sim q' \\ M &: H \centerdot (g \cdot L) \sim (f \cdot K) \centerdot H' \end{aligned}$$

Remark 10.1.5. The homotopy M witnesses 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. Therefore M is a homotopy of homotopies, and for each $z : C$ the identification $M(z)$ witnesses that the square of identifications

$$\begin{tikzcd}
f(p(z)) \arrow[r, equal, "\mathsf{ap}_f(K(z))"] \arrow[d, equal, "H(z)"'] & f(p'(z)) \arrow[d, equal, "H'(z)"] \\
g(q(z)) \arrow[r, equal, "\mathsf{ap}_g(L(z))"'] & g(q'(z))
\end{tikzcd}$$

commutes.

Proof of Lemma 10.1.4. By the fundamental theorem of identity types (Theorem 7.2.1) and associativity of Σ-types (Exercise 5.10) it suffices to show that the type

$$\textstyle\sum_{(p':C\to A)}\sum_{(q':C\to B)}\sum_{(H':f\circ p'\sim g\circ q')}\sum_{(K:p\sim p')}\sum_{(L:q\sim q')} H \centerdot (g\cdot L)\sim (f\cdot K)\centerdot H'$$

is contractible. Now we apply Exercise 5.11 repeatedly to see that this type is equivalent to the type

$$\textstyle\sum_{(p':C\to A)}\sum_{(K:p\sim p')}\sum_{(q':C\to B)}\sum_{(L:q\sim q')}\sum_{(H':f\circ p'\sim g\circ q')} H \centerdot (g\cdot L)\sim (f\cdot K)\centerdot H'.$$

The types $\sum_{(p':C\to A)} p\sim p'$ and $\sum_{(q':C\to B)} q\sim q'$ are contractible by function extensionality, and we have

$$\begin{aligned}(p,\mathsf{htpy.refl}_p) &: \textstyle\sum_{(p':C'\to A)} p\sim p'\\ (q,\mathsf{htpy.refl}_q) &: \textstyle\sum_{(q':C'\to B)} q\sim q'.\end{aligned}$$

Thus we apply Exercise 6.4 to see that the type of tuples (p', K, q', L, H', M) is equivalent to the type

$$\textstyle\sum_{(H':f\circ p'\sim g\circ q')} H \centerdot \mathsf{htpy.refl}_{g\circ q}\sim \mathsf{htpy.refl}_{f\circ p}\centerdot H'.$$

Of course, the type $H \centerdot \mathsf{htpy.refl}_{g\circ q}\sim \mathsf{htpy.refl}_{f\circ p}\centerdot H'$ is equivalent to the type $H\sim H'$, and $\sum_{(H':f\circ p\sim g\circ q)} H\sim H'$ is contractible. □

As a corollary we obtain the following characterization of the universal property of pullbacks.

Theorem 10.1.6. *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

$$\begin{aligned} h &: C' \to C \\ K &: p \circ h \sim p' \\ L &: q \circ h \sim q' \\ M &: (H \cdot h) \cdot (g \cdot L) \sim (f \cdot K) \cdot H' \end{aligned}$$

is contractible.

Remark 10.1.7. The homotopy M in Theorem 10.1.6 witnesses that the square

$$\begin{tikzcd} f \circ p \circ h \arrow[r, "f \cdot K"] \arrow[d, "H \cdot h"'] & f \circ p' \arrow[d, "H'"] \\ g \circ q \circ h \arrow[r, "g \cdot L"'] & g \circ q' \end{tikzcd}$$

of homotopies commutes.

10.2 The unique existence of pullbacks

Definition 10.2.1. Let $f : A \to X$ and $B \to X$ 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.

Theorem 10.2.2. *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

$$\mathsf{cone\_map}(\pi_1, \pi_2, \pi_3) : (C \to A \times_X B) \to \mathsf{cone}(C)$$

is an equivalence. By double application of Theorem 9.1.4 we obtain equivalences

$$\begin{aligned}(C \to A \times_X B) &\equiv C \to \textstyle\sum_{(x:A)} \sum_{(y:B)} f(x) = g(y) \\ &\simeq \textstyle\sum_{(p:C \to A)} \prod_{(z:C)} \sum_{(y:B)} f(p(z)) = y \\ &\simeq \textstyle\sum_{(p:C \to A)} \sum_{(q:C \to B)} \prod_{(z:C)} f(p(z)) = g(q(z)) \\ &\equiv \mathsf{cone}(C)\end{aligned}$$

The composite of these equivalences is the map

$$\lambda f.\, (\lambda z.\, \mathsf{pr}_1(f(z)), \lambda z.\, \mathsf{pr}_1(\mathsf{pr}_2(f(z))), \lambda z.\, \mathsf{pr}_2(\mathsf{pr}_2(f(z)))),$$

which is *exactly* the map $\mathsf{cone\_map}(\pi_1, \pi_2, \pi_3)$, and since it is a composite of equivalences it follows that it is itself an equivalence. □

In the following lemma we establish the uniqueness of pullbacks up to equivalence via a *3-for-2 property* for pullbacks.

Lemma 10.2.3. *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*

$$\begin{aligned} K &: p \circ h \sim p' \\ L &: q \circ h \sim q' \\ M &: (H \cdot h) \cdot (g \cdot L) \sim (f \cdot K) \cdot H'. \end{aligned}$$

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 10.1.4 we obtain an identification

$$\mathsf{cone\_map}((p,q,H),h) = (p',q',H')$$

from the triple (K,L,M). Let D be a type, and let $k : D \to C'$ be a map. We observe that

$$\begin{aligned}\mathsf{cone\_map}((p,q,H),(h\circ k)) &\equiv (p\circ(h\circ k), q\circ(h\circ k), H\circ(h\circ k))\\ &\equiv ((p\circ h)\circ k, (q\circ h)\circ k, (H\circ h)\circ k)\\ &\equiv \mathsf{cone\_map}(\mathsf{cone\_map}((p,q,H),h),k)\\ &= \mathsf{cone\_map}((p',q',H'),k).\end{aligned}$$

Thus we see that the triangle

$$\begin{tikzcd} (D \to C') \arrow[rr, "h\circ-"] \arrow[dr, "\mathsf{cone\_map}(p',q',H')"'] & & (D \to C) \arrow[dl, "\mathsf{cone\_map}(p,q,H)"] \\ & \mathsf{cone}(D) & \end{tikzcd}$$

commutes. Therefore it follows from the 3-for-2 property of equivalences established in Exercise 5.5, that if any two of the following properties hold, then so does the third:

(i) The map $\mathsf{cone\_map}(p,q,H) : (D \to C) \to \mathsf{cone}(D)$ is an equivalence,

(ii) The map $\mathsf{cone\_map}(p',q',H') : (D \to C') \to \mathsf{cone}(D)$ is an equivalence,

(iii) The map $h\circ - : (D \to C') \to (D \to C)$ is an equivalence.

Thus the 3-for-2 property for pullbacks 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 9.5. □

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 10.2.3 is contractible.

Corollary 10.2.4. *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

$$\begin{aligned} K &: p \circ e \sim p' \\ L &: q \circ e \sim q' \\ M &: (g \cdot L) \cdot (H \cdot e) \sim (f \cdot K) \cdot H'. \end{aligned}$$

is contractible.

Proof. We have seen that the type of quadruples (h, K, L, M) is equivalent to the fiber of $\mathsf{cone\_map}(p, q, H)$ at (p', q', H'). By Lemma 10.2.3 it follows that h is an equivalence. Since $\mathsf{is\_equiv}(h)$ is a proposition (and hence contractible as soon as it is inhabited) it follows that the type of quadruples (e, K, L, M) is contractible. □

Definition 10.2.5. 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 $\lambda z. (p(z), q(z), H(z))$. Furthermore, we will write

$$\mathsf{is\_pullback}(f, g, H) :\equiv \mathsf{is\_equiv}(\mathsf{gap}(p, q, H)).$$

Theorem 10.2.6. *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

$$\mathsf{is\_pullback}(p,q,H) :\equiv \mathsf{is\_equiv}(\mathsf{gap}(p,q,H)).$$

Proof. Note that there are homotopies

$$\begin{aligned} K &: \pi_1 \circ \mathsf{gap}(p,q,H) \sim p \\ L &: \pi_2 \circ \mathsf{gap}(p,q,H) \sim q \\ M &: (\pi_3 \cdot \mathsf{gap}(p,q,H)) \cdot (g \cdot L) \sim (f \cdot K) \cdot H. \end{aligned}$$

given by

$$\begin{aligned} K &:\equiv \lambda z.\, \mathsf{refl}_{p(z)} \\ L &:\equiv \lambda z.\, \mathsf{refl}_{q(z)} \\ M &:\equiv \lambda z.\, \mathsf{right\_unit}(H(z)) \cdot \mathsf{left\_unit}(H(z))^{-1}. \end{aligned}$$

Therefore the claim follows by Lemma 10.2.3. □

10.3 Fiber products

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 10.3.1. *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 10.2.6 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 10.3.1 is the reason why pullbacks are sometimes called **fiber products**.

Theorem 10.3.2. *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 10.2.6 it suffices to show that the gap map is an equivalence. The gap map is homotopic to the function

$$\lambda(x,(p,q)).\,((x,p),(x,q),\mathsf{refl}_x)$$

is an equivalence. The inverse of this function is the map

$$\lambda((x,p),(y,q),\alpha).\,(y,(\mathsf{tr}_P(\alpha,p),q)).$$ □

Corollary 10.3.3. *For any $f : A \to X$ and $g : B \to X$, the square*

$$\begin{tikzcd}
\sum_{(x:X)} \mathsf{fib}_f(x) \times \mathsf{fib}_g(y) \arrow[r, "\lambda(x,((a,p),(b,q))).\,b"] \arrow[d, "\lambda(x,((a,p),(b,q))).\,a"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"'] & X
\end{tikzcd}$$

is a pullback square.

10.4 Fibers as pullbacks

Lemma 10.4.1. *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 10.2.6 it suffices to show that the gap map is an equivalence. The gap map is homotopic to the function

$$\mathsf{total}(\lambda x.\, \lambda p.\, (\star, p))$$

The map $\lambda x.\, \lambda p.\, (\star, p)$ is a fiberwise equivalence by Exercise 6.4, so it induces an equivalence on total spaces by Theorem 7.1.3. □

Lemma 10.4.1 motivates the following definition of *fiber sequences*, which play an important role in synthetic homotopy theory (and in algebraic topology).

Definition 10.4.2. A **fiber sequence** consists of types F, E, and B with **base points** $x : F$, $y : E$, and $b : B$, and maps

$$F \xrightarrow{i} E \xrightarrow{p} B$$

preserving the base points in the sense that $i(x) = y$ and $p(y) = b$, such that the square

$$\begin{tikzcd} F \arrow[r, "i"] \arrow[d] & E \arrow[d, "p"] \\ \mathbf{1} \arrow[r, "b"'] & B \end{tikzcd}$$

is a pullback square. We often write $F \hookrightarrow E \twoheadrightarrow B$ to indicate that we have a fiber sequence.

Given a fiber sequence $F \hookrightarrow E \twoheadrightarrow B$, we call B the **base space**, E the **total space**, and F the **fiber**.

Example 10.4.3. 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.

To see this, note that the gap map is homotopic to the function

$$e :\equiv \lambda y.\, ((a, y), \mathsf{refl}_a).$$

This function is an equivalence by Exercise 7.5.

Thus we see that if we additionally suppose that there is a term $b : B(a)$, then we obtain a fiber sequence

$$B(a) \hookrightarrow \sum_{(x:A)} B(x) \twoheadrightarrow A.$$

10.5 Fiberwise equivalences

Lemma 10.5.1. *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 10.2.6 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 10.5.2. *Let $f : A \to B$, and let $g : \prod_{(a:A)} P(a) \to Q(f(a))$ be a fiberwise transformation. The following are equivalent:*

(i) The commuting square

$$\begin{tikzcd}
\sum_{(a:A)} P(a) \arrow[r, "\mathsf{total}_f(g)"] \arrow[d, two heads] & \sum_{(b:B)} Q(b) \arrow[d, two heads] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

is a pullback square.

(ii) g is a fiberwise equivalence.

Proof. The gap map is homotopic to the composite

$$\sum_{(x:A)} P(x) \xrightarrow{\mathsf{total}(g)} \sum_{(x:A)} Q(f(x)) \xrightarrow{\mathsf{gap}'} A \times_B \Big(\sum_{(y:B)} Q(y)\Big)$$

where gap' is the gap map for the square in Lemma 10.5.1. Since gap' is an equivalence, it follows by Exercise 5.5 and Theorem 7.1.3 that the gap map is an equivalence if and only if g is a fiberwise equivalence. □

Lemma 10.5.3. *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 $H : f \circ p \sim g \circ q$, and consider the fiberwise transformation

$$\mathsf{fib}_{(f,q,H)} : \prod\nolimits_{(a:A)} \mathsf{fib}_p(a) \to \mathsf{fib}_g(f(a))$$

given by $\lambda a.\, \lambda(c,u).\, (q(c), H(c)^{-1} \cdot \mathsf{ap}_f\,(u))$. Then there is an equivalence

$$\mathsf{fib}_{\mathsf{gap}(p,q,H)}((a,b,\alpha)) \simeq \mathsf{fib}_{\mathsf{fib}_{(f,q,H)}(a)}((b,\alpha^{-1}))$$

Proof. To obtain an equivalence of the desired type we simply concatenate known equivalences:

$$\begin{aligned} \mathsf{fib}_h((a,b,\alpha)) &\equiv \sum\nolimits_{(z:C)} (p(z), q(z), H(z)) = (a,b,\alpha) \\ &\simeq \sum\nolimits_{(z:C)} \sum\nolimits_{(u:p(z)=a)} \sum\nolimits_{(v:q(z)=b)} H(z) \cdot \mathsf{ap}_g\,(v) = \mathsf{ap}_f\,(u) \cdot \alpha \\ &\simeq \sum\nolimits_{((z,u):\mathsf{fib}_p(a))} \sum\nolimits_{(v:q(z)=b)} H(z)^{-1} \cdot \mathsf{ap}_f\,(u) = \mathsf{ap}_g\,(v) \cdot \alpha^{-1} \\ &\simeq \mathsf{fib}_{\varphi(a)}((b,\alpha^{-1})) \end{aligned}$$ □

Corollary 10.5.4. *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) The induced map on fibers

$$\lambda x.\, \lambda(z,\alpha).\, (q(z), H(z)^{-1} \cdot \mathsf{ap}_f\,(\alpha)) : \prod\nolimits_{(x:A)} \mathsf{fib}_p(x) \to \mathsf{fib}_g(f(x))$$

is a fiberwise equivalence.

Corollary 10.5.5. *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 Corollary 10.5.4 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 8.3.3, it follows that p is a k-truncated map. □

Corollary 10.5.6. *Consider a commuting square*

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X. \end{tikzcd}$$

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 10.5.2 the fibers of p are equivalent to the fibers of g, which are contractible by Theorem 6.3.3. Thus it follows that p is a contractible map, and hence that p is an equivalence.

If p is an equivalence, then by Theorem 6.3.3 both $\mathsf{fib}_p(x)$ and $\mathsf{fib}_g(f(x))$ are contractible for any $x : X$. It follows by Exercise 6.3 that the induced map $\mathsf{fib}_p(x) \to \mathsf{fib}_g(f(x))$ is an equivalence. Thus we apply Corollary 10.5.4 to conclude that the square is a pullback. □

Theorem 10.5.7. *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 fiberwise equivalences

$$\prod_{(x:X)} \mathsf{fib}_f(x) \simeq \mathsf{fib}_g(h(x)).$$

Corollary 10.5.8. *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*

$$i : \Big(\sum_{(x:X)} P(x)\Big) \to \Big(\sum_{(y:Y)} Q(y)\Big),$$

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 fiberwise equivalences

$$\prod_{(x:X)} P(x) \simeq Q(h(x)).$$

10.6 The pullback pasting property

The following theorem is also called the **pasting property** of pullbacks.

Theorem 10.6.1. *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 induces fiberwise transformations

$$\begin{aligned}&\textstyle\prod_{(x:X)}\mathsf{fib}_f(x) \to \mathsf{fib}_g(i(x))\\&\textstyle\prod_{(y:Y)}\mathsf{fib}_g(y) \to \mathsf{fib}_h(j(y)).\end{aligned}$$

By the assumption that the square on the right is a pullback square, it follows from Corollary 10.5.4 that the fiberwise transformation

$$\textstyle\prod_{(y:Y)}\mathsf{fib}_g(y) \to \mathsf{fib}_h(j(y))$$

is a fiberwise equivalence. Therefore it follows from 3-for-2 property of equivalences that the fiberwise transformation

$$\textstyle\prod_{(x:X)}\mathsf{fib}_f(x) \to \mathsf{fib}_g(i(x))$$

is a fiberwise equivalence if and only if the fiberwise transformation

$$\textstyle\prod_{(x:X)}\mathsf{fib}_f(x) \to \mathsf{fib}_h(j(i(x)))$$

is a fiberwise equivalence. Now the claim follows from one more application of Corollary 10.5.4. □

10.7 The disjointness of coproducts

As an application of the theory of pullbacks, we show that coproducts are disjoint. In this section we will write

$$[f,g] : A+B \to X$$

for the unique map satisfying $[f,g](\mathsf{inl}(x)) \equiv f(x)$ and $[f,g](\mathsf{inr}(y)) \equiv g(y)$, where $f : A \to X$ and $g : B \to X$. Furthermore, we will write

$$f+g :\equiv [\mathsf{inl}\circ f, \mathsf{inr}\circ g] : A+B \to X+B$$

for any $f : A \to X$ and $g : B \to Y$.

Lemma 10.7.1. *Let X be a type. Then we have the pullback squares*

$$\begin{tikzcd}
X \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{id}"'] & \mathbf{1} \arrow[d, "\mathsf{const}_{0_{\mathbf{2}}}"] & \mathbf{0} \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{const}_{1_{\mathbf{2}}}"] \\
X \arrow[r, "\mathsf{const}_{0_{\mathbf{2}}}"'] & \mathbf{2} & X \arrow[r, "\mathsf{const}_{0_{\mathbf{2}}}"'] & \mathbf{2},
\end{tikzcd}$$

and we have similar pullback squares with the roles of $0_{\mathbf{2}}$ and $1_{\mathbf{2}}$ reversed.

Proof. For the first square we observe that both squares and the outer rectangle in the diagram

$$\begin{tikzcd}
X \arrow[r] \arrow[d] & \mathbf{1} \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{const}_{0_{\mathbf{2}}}"] \\
X \arrow[r, "\mathsf{const}_\star"'] & \mathbf{1} \arrow[r, "\mathsf{const}_{0_{\mathbf{2}}}"'] & \mathbf{2}.
\end{tikzcd}$$

are pullback squares. To see this, recall that the identity type $0_{\mathbf{2}} = 0_{\mathbf{2}}$ is contractible by Exercise 8.3. Therefore it follows that the square on the right is a pullback square by Exercise 10.1. The square on the left is a pullback square by Corollary 10.5.6. Therefore the outer rectangle is a pullback square by Theorem 10.6.1.

For the second square we observe that both squares end the outer rectangle in the diagram

$$\begin{tikzcd}
\mathbf{0} \arrow[r] \arrow[d] & \mathbf{0} \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{const}_{1_{\mathbf{2}}}"] \\
X \arrow[r, "\mathsf{const}_\star"'] & \mathbf{1} \arrow[r, "\mathsf{const}_{0_{\mathbf{2}}}"'] & \mathbf{2}.
\end{tikzcd}$$

are pullback squares. To see this, recall that the identity type $0_{\mathbf{2}} = 1_{\mathbf{2}}$ is equivalent to the empty type by Exercise 8.3. Therefore it follows that the square on the right is a pullback. It is also straightforward to verify that the square on the left is a pullback. Therefore it follows from Theorem 10.6.1 that the outer rectangle is a pullback. □

Lemma 10.7.2. *For any two types A and B, the squares*

$$\begin{tikzcd}
A \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{inl}"'] & \mathbf{1} \arrow[d, "\mathsf{const}_{0_{\mathbf{2}}}"] & B \arrow[r, "\mathsf{const}_\star"] \arrow[d, "\mathsf{inr}"'] & \mathbf{1} \arrow[d, "\mathsf{const}_{1_{\mathbf{2}}}"] \\
A+B \arrow[r, "{[\mathsf{const}_{0_{\mathbf{2}}},\mathsf{const}_{1_{\mathbf{2}}}]}"'] & \mathbf{2} & A+B \arrow[r, "{[\mathsf{const}_{0_{\mathbf{2}}},\mathsf{const}_{1_{\mathbf{2}}}]}"'] & \mathbf{2}
\end{tikzcd}$$

are pullback squares.

Proof. The two cases are similar, so we only give the proof that the left square is a pullback. The left square commutes by the homotopy

$$H :\equiv \mathsf{htpy.refl}_{\mathsf{const}_{0_{\mathbf{2}}}}.$$

To see that the asserted square is a pullback square we use Theorem 10.2.6 and show that the gap map is an equivalence. First we note that the gap map is homotopic to the function $e : A \to (A+B) \times_{\mathbf{2}} \mathbf{1}$ is defined by

$$\lambda x.\, (\mathsf{inl}(x), \star, \mathsf{refl}_{0_{\mathbf{2}}}).$$

The inverse is defined by the induction principle of coproducts by

$$\begin{aligned} e^{-1}(\mathsf{inl}(x), t, \alpha) &:\equiv x \\ e^{-1}(\mathsf{inr}(y), t, \alpha) &:\equiv \mathsf{ind}_{\mathbf{0}}(\zeta(\alpha)), \end{aligned}$$

where $\zeta : \prod_{(x,y:\mathbf{2})}(x = y) \to \mathsf{Eq}_{\mathbf{2}}(x,y)$ is the canonical map of the identity type of $\mathbf{2}$ into the observational equality on $\mathbf{2}$. In the case of $\alpha : 0_{\mathbf{2}} = 1_{\mathbf{2}}$ we obtain a term of $\mathsf{Eq}_{\mathbf{2}}(0_{\mathbf{2}}, 1_{\mathbf{2}}) \equiv \mathbf{0}$. It is immediate from the computation rules that $e^{-1} \circ e \equiv \mathsf{id}$.

The homotopy $e \circ e^{-1} \sim \mathsf{id}$ is again constructed by the induction principle of coproducts. In the inl-case we have $e(e^{-1}(\mathsf{inl}(x), t, \alpha)) \equiv (\mathsf{inl}(x), \star, \mathsf{refl}_{0_{\mathbf{2}}})$. We define the identification

$$(\mathsf{inl}(x), \star, \mathsf{refl}_{0_{\mathbf{2}}}) = (\mathsf{inl}(x), t, \alpha)$$

by singleton induction on $t : \mathbf{1}$ and $\alpha : 0_{\mathbf{2}} = 0_{\mathbf{2}}$ (both of which are terms of contractible types). Thus, it suffices to provide an identification

$$(\mathsf{inl}(x), \star, \mathsf{refl}_{0_{\mathbf{2}}}) = (\mathsf{inl}(x), \star, \mathsf{refl}_{0_{\mathbf{2}}}),$$

which we have by reflexivity. The inr-case is again automatic, since we obtain a term of the empty type from $\alpha : 0_{\mathbf{2}} = 1_{\mathbf{2}}$. This completes the proof that e is an equivalence. □

Corollary 10.7.3. *The maps* $\mathsf{inl} : A \to A + B$ *and* $\mathsf{inr} : B \to A + B$ *are embeddings.*

Proof. By the pullback squares of Lemma 10.7.2 and Corollary 10.5.5 it suffices to show that $\mathbf{1} \to \mathbf{2}$ is an embedding. This is Exercise 8.9. □

Theorem 10.7.4. *Coproducts are* ***disjoint*** *in the sense that for any two types A and B, the commuting square*

$$\begin{tikzcd}
\mathbf{0} \arrow[r] \arrow[d] & B \arrow[d, "\mathsf{inr}"] \\
A \arrow[r, "\mathsf{inl}"'] & A+B
\end{tikzcd}$$

is a pullback square.

Proof. Now consider the commuting diagram

$$\begin{tikzcd}
\mathbf{0} \arrow[r] \arrow[d] & B \arrow[r] \arrow[d, "\mathsf{inr}"] & \mathbf{1} \arrow[d, "\mathsf{const}_{1_{\mathbf{2}}}"] \\
A \arrow[r, "\mathsf{inl}"'] & A+B \arrow[r, "{[\mathsf{const}_{0_{\mathbf{2}}},\mathsf{const}_{1_{\mathbf{2}}}]}"'] & \mathbf{2}.
\end{tikzcd}$$

By Lemma 10.7.1 it follows that the outer rectangle is a pullback square. The square on the right is a pullback square by Lemma 10.7.2. Therefore the square on the left is a pullback square by Theorem 10.6.1. □

Corollary 10.7.5. *Let A and B be types. There are equivalences*

$$\begin{aligned}
(\mathsf{inl}(x) = \mathsf{inl}(x')) &\simeq (x =_A x') \\
(\mathsf{inl}(x) = \mathsf{inr}(y')) &\simeq \mathbf{0} \\
(\mathsf{inr}(y) = \mathsf{inl}(x')) &\simeq \mathbf{0} \\
(\mathsf{inr}(y) = \mathsf{inr}(y')) &\simeq (y =_B y').
\end{aligned}$$

Proof. The cases

$$\begin{aligned}
(\mathsf{inl}(x) = \mathsf{inl}(x')) &\simeq (x =_A x') \\
(\mathsf{inr}(y) = \mathsf{inr}(y')) &\simeq (y =_B y').
\end{aligned}$$

follow from Corollary 10.7.3 since both inl and inr are embeddings. The remaining cases follow from the disjointness of coproducts, proven in Theorem 10.7.4. □

Exercises

10.1 (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 8.1.

10.2 In this exercise we give an alternative characterization of the notion of k-truncated map, compared to Theorem 8.3.7. 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.

10.3 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 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.

10.4 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 $\lambda z.\,\mathsf{eq\_pair}(H(z), \mathsf{refl}_{g(q(z))})$ is a pullback square.

10.5 Show that if

$$\begin{tikzcd}
C_1 \arrow[r] \arrow[d] & B_1 \arrow[d] \\
A_1 \arrow[r] & X_1
\end{tikzcd}
\qquad
\begin{tikzcd}
C_2 \arrow[r] \arrow[d] & B_2 \arrow[d] \\
A_2 \arrow[r] & X_2
\end{tikzcd}$$

are pullback squares, then so is

$$\begin{tikzcd}
C_1 \times C_2 \arrow[r] \arrow[d] & B_1 \times B_2 \arrow[d] \\
A_1 \times A_2 \arrow[r] & X_1 \times X_2.
\end{tikzcd}$$

10.6 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 $H_i : f_i \circ p_i \sim g_i \circ q_i$.

(a) Show that the square

$$\begin{tikzcd}
\sum_{(i:I)} C_i \arrow[r, "\mathsf{total}(q)"] \arrow[d, "\mathsf{total}(p)"'] & \sum_{(i:I)} B_i \arrow[d, "\mathsf{total}(g)"] \\
\sum_{(i:I)} A_i \arrow[r, "\mathsf{total}(f)"'] & \sum_{(i:I)} X_i
\end{tikzcd}$$

which commutes by the homotopy

$$\mathsf{total}(H) :\equiv \lambda(i,c).\,\mathsf{eq\_pair}(\mathsf{refl}_i, H_i(c))$$

is a pullback square.

(b) 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.

10.7 Let B be a type family over A. Show that the square

$$\begin{tikzcd}
\prod_{(x:A)} B(x) \arrow[r, "\lambda f.\,\lambda x.\,(x,f(x))"] \arrow[d] & \Big(\sum_{(x:A)} B(x)\Big)^A \arrow[d, "\mathsf{pr}_1\circ -"] \\
\mathbf{1} \arrow[r, "\mathsf{const}_{\mathsf{id}_A}"'] & A^A
\end{tikzcd}$$

is a pullback square. Conclude that the type $\prod_{(x:A)} B(x)$ is equivalent to the type $\mathsf{sec}(\mathsf{pr}_1)$ of sections of the projection map.

10.8 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 $c_1, c_2 : C$. Show that the square

$$\begin{tikzcd} (c_1 = c_2) \arrow[r, "\mathsf{ap}_q"] \arrow[d, "\mathsf{ap}_p"'] & (q(c_1) = q(c_2)) \arrow[d, "\lambda\beta.\, H(c_1) \cdot \mathsf{ap}_g(\beta)"] \\ (p(c_1) = p(c_2)) \arrow[r, "\lambda\alpha.\, \mathsf{ap}_f(\alpha) \cdot H(c_2)"'] & f(p(c_1)) = g(q(c_2)), \end{tikzcd}$$

which commutes by the naturality of homotopies (Definition 6.3.1), is again a pullback square.

10.9 Suppose that

$$\begin{tikzcd} C \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & X \end{tikzcd}$$

with $H : f \circ p \sim g \circ q$ is a pullback square. Show that 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 again a pullback square.

10.10 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 $H : f \circ p \sim g \circ q$, and let $h : C \to A \times_X B$ be the map given by $h(z) :\equiv (p(c), q(c), H(c))$. Show that the square

$$\begin{tikzcd} \mathsf{fib}_{\mathsf{gap}(p,q,H)}((a,b,\alpha)) \arrow[r, "\lambda(c,\beta).\,(c,\mathsf{ap}_{\pi_1}(\beta))"] \arrow[d, "\mathsf{const}_\star"'] & \mathsf{fib}_p(a) \arrow[d, "\mathsf{fib}_{(f,g,H)}"] \\ \mathbf{1} \arrow[r, "\mathsf{const}_{(b,\alpha^{-1})}"'] & \mathsf{fib}_g(f(a)) \end{tikzcd}$$

10.11 Consider a commuting diagram of the form

$$\begin{tikzcd}
A_0 \arrow[r] \arrow[d] & B_0 \arrow[d] & C_0 \arrow[l] \arrow[d] \\
A_1 \arrow[r] & B_1 & C_1 \arrow[l] \\
A_2 \arrow[r] \arrow[u] & B_2 \arrow[u] & C_2 \arrow[l] \arrow[u]
\end{tikzcd}$$

with homotopies filling the (small) squares. Construct an equivalence

$$\begin{aligned}
&(A_0 \times_{B_0} C_0) \times_{(A_1 \times_{B_1} C_1)} (A_2 \times_{B_2} C_2) \\
&\qquad \simeq (A_0 \times_{A_1} A_2) \times_{(B_0 \times_{B_1} B_2)} (C_0 \times_{C_1} C_2).
\end{aligned}$$

This is also known as the **3-by-3 lemma** for pullbacks.

Lecture 11

The univalence axiom

11.1 Type extensionality

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. In the following theorem we introduce the univalence axiom and give two more equivalent ways of stating this.

Theorem 11.1.1. *The following are equivalent:*

(i) The ***univalence axiom****: for any $A : \mathcal{U}$ the map*

$$\mathsf{equiv\_eq} :\equiv \mathsf{ind}_{A=}(\mathsf{id}_A) : \textstyle\prod_{(B:\mathcal{U})}(A = B) \to (A \simeq B).$$

is a fiberwise equivalence. If this is the case, we write equiv_eq *for the inverse of* equiv_eq*.*

(ii) The type

$$\textstyle\sum_{(B:\mathcal{U})} A \simeq B$$

is contractible for each $A : \mathcal{U}$.

(iii) The principle of ***equivalence induction****: for every $A : \mathcal{U}$ and for every type family*

$$P : \textstyle\prod_{(B:\mathcal{U})}(A \simeq B) \to \mathsf{Type},$$

the map

$$\Big(\textstyle\prod_{(B:\mathcal{U})}\prod_{(e:A\simeq B)} P(B,e)\Big) \to P(A, \mathsf{id}_A)$$

given by $f \mapsto f(A, \mathsf{id}_A)$ has a section.

Proof. The equivalence of (i) and (ii) is a direct consequence of Theorem 7.2.1. To see that (ii) and (iii) are equivalent, note that we have a commuting triangle

$$\begin{tikzcd}
\prod_{(t:\sum_{(B:\mathcal{U})} A\simeq B)} P(t) \arrow[rr, "\mathsf{ind}_\Sigma"] \arrow[dr, "\varphi", "f\mapsto f((A,\mathsf{id}_A))"'] & & \prod_{(B:\mathcal{U})}\prod_{(e:A\simeq B)} P((B,e)) \arrow[dl, "\psi"', "f\mapsto f(A,\mathsf{id}_A)"] \\
& P((A,\mathsf{id}_A)) &
\end{tikzcd}$$

The map ind_Σ has a section. Therefore it follows from Exercise 5.5 that φ has a section if and only if ψ has a section. By Theorem 6.1.1 it follows that φ has a section if and only if $\sum_{(B:\mathcal{U})} A \simeq B$ is contractible. □

From now on we will assume that the univalence axiom holds.

11.2 Groups in univalent mathematics

In this section we exhibit 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 [2]. 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].

Definition 11.2.1. A **group** $\mathcal{G}$ consists of a type G equipped with

$$\begin{aligned}
p &: \mathsf{is\_set}(G) \\
1 &: G \\
i &: G \to G \\
\mu &: G \to (G \to G),
\end{aligned}$$

satisfying the **group laws**:

$$\begin{aligned}
\mathsf{assoc} &: \textstyle\prod_{(x,y,z:G)} \mu(\mu(x,y),z) = \mu(x,\mu(y,z)) \\
\mathsf{left\_unit} &: \textstyle\prod_{(x:G)} \mu(1,x) = x \\
\mathsf{right\_unit} &: \textstyle\prod_{(x:G)} \mu(x,1) = x \\
\mathsf{left\_inv} &: \textstyle\prod_{(x:G)} \mu(i(x),x) = 1
\end{aligned}$$

[1] A notable exception is that of categories, which are objects at truncation level 1, i.e. at the level of *groupoids*. For more on this, see Chapter 9 of [2].

$$\mathsf{right\_inv} : \textstyle\prod_{(x:G)}\mu(x, i(x)) = x.$$

The type Grp of all small groups is defined as

$$\begin{aligned}\mathsf{Grp} :\equiv \textstyle\sum_{(G:\mathcal{U})}\sum_{(p:\mathsf{is\_set}(G))}\sum_{(1:G)}\sum_{(i:G\to G)}\sum_{(\mu:G\to(G\to G))}\\ \Big(\textstyle\prod_{(x,y,z:G)}\mu(\mu(x,y),z) = \mu(x,\mu(y,z))\Big)\times\\ \Big(\textstyle\prod_{(x:G)}\mu(1,x) = x\Big) \times \Big(\textstyle\prod_{(x:G)}\mu(x,1) = x\Big)\times\\ \Big(\textstyle\prod_{(x:G)}\mu(i : x, x) = 1\Big) \times \Big(\textstyle\prod_{(x:G)}\mu(x, i(x)) = x\Big).\end{aligned}$$

We will usually write x^{-1} for $i(x)$, and xy for $\mu(x, y)$. The binary operation μ is also referred to as the **group operation**.

Example 11.2.2. An important class of examples consists of the **loop space** $x = x$ 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 §4.2.

Using higher inductive types we will show in Lecture 16 that *every* group is of this form.

Example 11.2.3. 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 8.2.4 and Exercise 8.4. The group laws were shown in Exercise 5.12.

Definition 11.2.4. Let $\mathcal{G}$ and $\mathcal{G}'$ be groups. The type $\hom(\mathcal{G}, \mathcal{G}')$ of **group homomorphisms** from $\mathcal{G}$ to $\mathcal{G}'$ is defined to be the type of pairs (h, p) consisting of

$$\begin{aligned}h &: G \to G'\\ p &: \textstyle\prod_{(x,y:G)} f(xy) = f(x)f(y).\end{aligned}$$

Remark 11.2.5. Since preservation of the group operation is a property, we will usually write h for the group homomorphism (h, p). Moreover, from Corollary 8.3.9 we obtain that the projection map

$$\mathsf{pr}_1 : \hom(\mathcal{G}, \mathcal{G}') \to (G \to G')$$

is an embedding. Therefore the equality type $(h,p)=(h',p')$ is equivalent to $h=h'$. In other words, to show that two group homomorphisms are equal it suffices to show that their underlying maps are equal.

Lemma 11.2.6. *For any two groups $\mathcal{G}$, and $\mathcal{H}$, the type* $\mathrm{hom}(\mathcal{G},\mathcal{H})$ *is equivalent to the type of quadruples (f,α,β,γ) consisting of*

$$\begin{aligned}
&f : G \to H\\
&\alpha : f(1)=1\\
&\beta : \textstyle\prod_{(x:G)} f(x^{-1})=f(x)^{-1}\\
&\gamma : \textstyle\prod_{(x,y:G)} f(xy)=f(x)f(y).
\end{aligned}$$

Proof. It suffices to show that for any group homomorphism $f:\mathrm{hom}(\mathcal{G},\mathcal{H})$, the types $f(1)=1$ and

$$\Big(\textstyle\prod_{(x:G)} f(x^{-1})=f(x)^{-1}\Big)$$

are contractible. Since G is a set, both types are propositions. Therefore it suffices to show they are inhabited. In other words, it suffices to show that any group homomorphism preserves the unit element and inverses. These are just calculations, where each step is an application of a group law:

$$\begin{aligned}
f(1) &= 1f(1) & f(x^{-1}) &= f(x^{-1})1\\
&= (f(1)^{-1}f(1))f(1) & &= f(x^{-1})(f(x)f(x)^{-1})\\
&= f(1)^{-1}(f(1)f(1)) & &= (f(x^{-1})f(x))f(x)^{-1}\\
&= f(1)^{-1}f(11) & &= f(x^{-1}x)f(x)^{-1}\\
&= f(1)^{-1}f(1) & &= f(1)f(x)^{-1}\\
&= 1. & &= 1f(x)^{-1}\\
& & &= f(x)^{-1}. \qquad \square
\end{aligned}$$

Definition 11.2.7. Let $\mathcal{G}$ be a group. Then the **identity homomorphism** $\mathsf{id}_{\mathcal{G}}:\mathrm{hom}(\mathcal{G},\mathcal{G})$ is defined to be the pair (id_G,p), where

$$p(x,y) :\equiv \mathsf{refl}_{xy}.$$

Definition 11.2.8. Let $h:\mathrm{hom}(\mathcal{G},\mathcal{H})$ and $k:\mathrm{hom}(\mathcal{H},\mathcal{K})$ be group homomorphisms, with proofs p and q that h and k preserve the group operation, respectively. Then we define

$$k\circ h : \mathrm{hom}(\mathcal{G},\mathcal{K})$$

to be the group homomorphism with underlying map $k \circ h$. This map preserves the group operation since

$$k(h(xy)) = k(h(x)h(y)) = k(h(x))k(h(y)).$$

Definition 11.2.9. Let $h : \hom(\mathcal{G}, \mathcal{H})$ be a group homomorphism. Then h is said to be an **isomorphism** if there is a group homomorphism $h^{-1} : \hom(\mathcal{H}, \mathcal{G})$ such that

$$h^{-1} \circ h = \mathsf{id}_{\mathcal{G}} \qquad \text{and} \qquad h \circ h^{-1} = \mathsf{id}_{\mathcal{H}}.$$

We write $\mathcal{G} \cong \mathcal{H}$ for the type of all group isomorphisms from $\mathcal{G}$ to $\mathcal{H}$, i.e.

$$\mathcal{G} \cong \mathcal{H} :\equiv \textstyle\sum_{(h:\hom(\mathcal{G},\mathcal{H}))} \sum_{(k:\hom(\mathcal{H},\mathcal{G}))} (k \circ h = \mathsf{id}_{\mathcal{G}}) \times (h \circ k = \mathsf{id}_{\mathcal{H}}).$$

Lemma 11.2.10. *The type of isomorphisms $\mathcal{G} \cong \mathcal{H}$ is equivalent to the type*

$$\begin{aligned} e &: G \simeq H \\ \alpha &: e(1) = 1 \\ \beta &: \textstyle\prod_{(x:G)} e(x^{-1}) = e(x)^{-1} \\ \gamma &: \textstyle\prod_{(x,y:G)} e(xy) = e(x)e(y). \end{aligned}$$

Proof. 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. Since being a group homomorphism is a property, it follows that the type of group isomorphism is equivalent to the type of group homomorphisms of which the underlying map has an inverse. By Exercise 9.14 it follows that the type

$$\textstyle\sum_{(f:\hom(\mathcal{G},\mathcal{H}))} \mathsf{is\_invertible}(f)$$

of group homomorphism of which the underlying map has an inverse is equivalent to the type

$$\textstyle\sum_{(f:\mathsf{hom}(\mathcal{G},\mathcal{H}))} \mathsf{is\_equiv}(f).$$

of group homomorphisms of which the underlying map is an equivalence. □

Definition 11.2.11. Let $\mathcal{G} : \mathsf{Grp}$ be a group. We define the map

$$\mathsf{iso\_eq} : \textstyle\prod_{(\mathcal{H}:\mathsf{Grp})} (\mathcal{G} = \mathcal{H}) \to (\mathcal{G} \cong \mathcal{H})$$

by path induction, taking $\mathsf{refl}_{\mathcal{G}}$ to $\mathsf{id}_{\mathcal{G}}$. Indeed, $\mathsf{id}_{\mathcal{G}}$ is a group isomorphism since it is its own inverse.

Theorem 11.2.12. *The fiberwise transformation*

$$\mathsf{iso\_eq} : \textstyle\prod_{(\mathcal{G}':\mathsf{Grp})}(\mathcal{G} = \mathcal{G}') \to (\mathcal{G} \cong \mathcal{G}')$$

is a fiberwise equivalence, for any group $\mathcal{G}$.

Proof. We will apply Theorem 7.2.1, and show that the type

$$\textstyle\sum_{(\mathcal{G}':\mathsf{Grp})}\mathcal{G} \cong \mathcal{G}'$$

is contractible. By Lemma 11.2.10 it follows that the total space $\sum_{(\mathcal{G}':\mathsf{Grp})}(\mathcal{G} \cong \mathcal{G}')$ is equivalent to the type

$$\begin{aligned}&\textstyle\sum_{(G':\mathcal{U})}\sum_{(p':\mathsf{is\_set}(G'))}\\ &\qquad\textstyle\sum_{(1':G)}\sum_{(i':G'\to G')}\sum_{(\mu':G'\to(G'\to G'))}\sum_{(L':\mathsf{group\_laws}(G',1',i',\mu'))}\\ &\qquad\qquad\textstyle\sum_{(e:G\simeq G')}\Big(e(1) = 1'\Big) \times \Big(\prod_{(x:G)}e(x^{-1}) = i'(e(x))\Big)\times\\ &\qquad\qquad\qquad\Big(\textstyle\prod_{(x,y:G)}e(xy) = \mu'(e(x), e(y))\Big).\end{aligned}$$

By the univalence axiom, the type $\sum_{(G':\mathcal{U})} G \simeq G'$ is contractible. Thus we see that the above type is equivalent to

$$\begin{gathered}\textstyle\sum_{(q:\mathsf{is\_set}(G))}\sum_{(1':G)}\sum_{(i':G\to G)}\sum_{(\mu':G\to(G\to G))}\sum_{(L:\mathsf{group\_laws}(G,1',i',\mu'))}\\ (1 = 1') \times \Big(\textstyle\prod_{(x:G)}x^{-1} = i'(x)\Big) \times \Big(\prod_{(x,y:G)}xy = \mu'(x, y)\Big).\end{gathered}$$

Of course, the types

$$\begin{aligned}&\textstyle\sum_{(1':G)}1 = 1'\\ &\textstyle\sum_{(i':G\to G)}\prod_{(x:G)}x^{-1} = i'(x)\\ &\textstyle\sum_{(\mu':G\to(G\to G))}\Big(\prod_{(x,y:G)}xy = \mu'(x, y)\Big)\end{aligned}$$

are all contractible. Moreover, being a set is a proposition, and since G is a set the group laws are propositions too. Since G is already assumed to be a set on which the group operations satisfy the group laws, it follows that the types $\mathsf{is\_set}(G)$ and $\mathsf{group\_laws}(G, 1, i, \mu)$ are all contractible. This concludes the proof that the total space $\sum_{(\mathcal{G}':\mathsf{Grp})} \mathcal{G} \cong \mathcal{G}'$ is contractible. □

Corollary 11.2.13. *The type* Grp *is a* 1*-type.*

Proof. It is straightforward to see that the type of group isomorphisms $\mathcal{G} \cong \mathcal{H}$ is a set, for any two groups $\mathcal{G}$ and $\mathcal{H}$. □

11.3 Equivalence relations

Definition 11.3.1. Let $R : A \to (A \to \mathsf{Prop})$ be a binary relation valued in the propositions. We say that R is an **(0-)equivalence relation** if R comes equipped with

$$\begin{aligned}
&\rho : \textstyle\prod_{(x:A)} R(x,x) \\
&\sigma : \textstyle\prod_{(x,y:A)} R(x,y) \to R(y,x) \\
&\tau : \textstyle\prod_{(x,y,z:A)} R(x,y) \to (R(y,z) \to R(x,z)).
\end{aligned}$$

Given an equivalence relation $R : A \to (A \to \mathsf{Prop})$, the **equivalence class** $[x]_R$ of $x : A$ is defined to be

$$[x]_R :\equiv R(x).$$

Definition 11.3.2. Let $R : A \to (A \to \mathsf{Prop})$ be a 0-equivalence relation. We define for any $x, y : A$ a map

$$\mathsf{class\_eq} : R(x,y) \to ([x]_R = [y]_R).$$

Construction. Let $r : R(x,y)$. By function extensionality, the identity type $R(x) = R(y)$ is equivalent to the type

$$\textstyle\prod_{(z:A)} R(x,z) = R(y,z).$$

Let $z : A$. By the univalence axiom, the type $R(x,z) = R(y,z)$ is equivalent to the type

$$R(x,z) \simeq R(y,z).$$

We have the map $\tau_{y,x,z}(\sigma(r)) : R(x,z) \to R(y,z)$. Since this is a map between propositions, we only have to construct a map in the converse direction to show that it is an equivalence. The map in the converse direction is just $\tau_{x,y,z}(r) : R(y,z) \to R(x,z)$. □

Theorem 11.3.3. *Let $R : A \to (A \to \mathsf{Prop})$ be a 0-equivalence relation. Then for any $x, y : A$ the map*

$$\mathsf{class\_eq} : R(x,y) \to ([x]_R = [y]_R)$$

is an equivalence.

Proof. By the 3-for-2 property of equivalences, it suffices to show that the map

$$\lambda r.\, \lambda z.\, \tau_{y,x,z}(\sigma(r)) : R(x,y) \to \prod_{(z:A)} R(x,z) \simeq R(y,z)$$

is an equivalence. Since this is a map between propositions, it suffices to construct a map of type

$$\Big(\prod_{(z:A)} R(x,z) \simeq R(y,z)\Big) \to R(x,y).$$

This map is simply $\lambda f.\, \sigma_{y,x}(f_x(\rho(x)))$. □

Remark 11.3.4. By Theorem 11.3.3 we can begin to think of the *quotient* A/R of a type A by an equivalence relation R. Classically, the quotient is described as the set of equivalence classes, and Theorem 11.3.3 establishes that two equivalence classes $[x]_R$ and $[y]_R$ are equal precisely when x and y are related by R.

However, the type $A \to \mathsf{Prop}$ may contain many more terms than just the R-equivalence classes. Therefore we are facing the task of finding a type theoretic description of the smallest subtype of $A \to \mathsf{Prop}$ containing the equivalence classes. Another to think about this is as the *image* of R in $A \to \mathsf{Prop}$. The construction of the (homotopy) image of a map can be done with *higher inductive types*, which we will do in Lecture 16.

The notion of 0-equivalence relation which we defined in Definition 11.3.1 fits in a hierarchy of 'n-equivalence relations', the study of which is a research topic on its own. However, we already know an example of a relation that should count as an '∞-equivalence relation': the identity type. Analogous to Theorem 11.3.3, the following theorem shows that the canonical map

$$(x = y) \to (\mathsf{Id}_A(x) = \mathsf{Id}_A(y))$$

is an equivalence, for any $x, y : A$. In other words, $\mathsf{Id}_A(x)$ can be thought of as the equivalence class of x with respect to the relation Id_A.

Theorem 11.3.5. *Assuming the univalence axiom on $\mathcal{U}$, the map*

$$\mathsf{Id}_A : A \to (A \to \mathcal{U})$$

is an embedding, for any type $A : \mathcal{U}$.

Proof. Let $a : A$. By function extensionality it suffices to show that the canonical map

$$(a = b) \to \mathsf{Id}_A(a) \sim \mathsf{Id}_A(b)$$

that sends refl_a to $\lambda x.\,\mathsf{refl}_{(a=x)}$ is an equivalence for every $b : A$, and by univalence it therefore suffices to show that the canonical map

$$(a = b) \to \prod_{(x:A)}(a = x) \simeq (b = x)$$

that sends refl_a to $\lambda x.\,\mathsf{id}_{(a=x)}$ is an equivalence for every $b : B$. To do this we employ the type theoretic Yoneda lemma, Theorem 9.2.3.

By the type theoretic Yoneda lemma we have an equivalence

$$\Big(\prod_{(x:A)}(b = x) \to (a = x)\Big) \to (a = b)$$

given by $\lambda f.\, f(b, \mathsf{refl}_b)$, for every $b : A$. Note that any fiberwise map $\prod_{(x:A)}(b = x) \to (a = x)$ induces an equivalence of total spaces by Exercise 6.3, since their total spaces are are both contractible by Corollary 6.3.4. It follows that we have an equivalence

$$\varphi_b : \Big(\prod_{(x:A)}(b = x) \simeq (a = x)\Big) \to (a = b)$$

given by $\lambda f.\, f(b, \mathsf{refl}_b)$, for every $b : A$.

Note that $\varphi_a(\lambda x.\,\mathsf{id}_{(a=x)}) \equiv \mathsf{refl}_a$. Therefore it follows by another application of Theorem 9.2.3 that the unique family of maps

$$\alpha_b : (a = b) \to \Big(\prod_{(x:A)}(b = x) \simeq (a = x)\Big)$$

that satisfies $\alpha_a(\mathsf{refl}_a) = \lambda x.\,\mathsf{id}_{(a=x)}$ is a fiberwise section of φ. It follows that α is a fiberwise equivalence. Now the proof is completed by reverting the direction of the fiberwise equivalences in the codomain. □

11.4 Essentially small types and maps

It is a trivial observation, but nevertheless of fundamental importance, that by the univalence axiom the identity types of $\mathcal{U}$ are equivalent to types in $\mathcal{U}$, because it provides an equivalence $(A = B) \simeq (A \simeq B)$, and the type $A \simeq B$ is in $\mathcal{U}$ for any $A, B : \mathcal{U}$. Since the identity types of $\mathcal{U}$ are equivalent to types in $\mathcal{U}$, we also say that the universe is *locally small*.

Definition 11.4.1. (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

$$\mathsf{ess\_small}(A) :\equiv \sum_{(X:\mathcal{U})} A \simeq X.$$

(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

$$\mathsf{ess\_small}(f) :\equiv \textstyle\prod_{(b:B)} \mathsf{ess\_small}(\mathsf{fib}_f(b)).$$

(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

$$\mathsf{loc\_small}(A) :\equiv \textstyle\prod_{(x,y:A)} \mathsf{ess\_small}(x = y).$$

Lemma 11.4.2. *The type* $\mathsf{ess\_small}(A)$ *is a proposition for any type* A.

Proof. Let X be a type. Our goal is to show that the type

$$\textstyle\sum_{(Y:\mathcal{U})} X \simeq Y$$

is a proposition. Suppose there is a type $X' : \mathcal{U}$ and an equivalence $e : X \simeq X'$, then the map

$$(X \simeq Y) \to (X' \simeq Y)$$

given by precomposing with e^{-1} is an equivalence. This induces an equivalence on total spaces

$$\Big(\textstyle\sum_{(Y:\mathcal{U})} X \simeq Y\Big) \simeq \Big(\textstyle\sum_{(Y:\mathcal{U})} X' \simeq Y\Big)$$

However, the codomain of this equivalence is contractible by Theorem 11.1.1. Thus it follows by Corollary 8.1.4 that the asserted type is a proposition. □

Corollary 11.4.3. *For each function* $f : A \to B$*, the type* $\mathsf{ess\_small}(f)$ *is a proposition, and for each type* X *the type* $\mathsf{loc\_small}(X)$ *is a proposition.*

Proof. This follows from the fact that propositions are closed under dependent products, established in Theorem 9.1.2. □

Theorem 11.4.4. *For any small type* $A : \mathcal{U}$ *there is an equivalence*

$$\mathsf{map\_fam}_A : (A \to \mathcal{U}) \simeq \Big(\textstyle\sum_{(X:\mathcal{U})} X \to A\Big).$$

Proof. Note that we have the function

$$\varphi : \lambda B. \Big(\textstyle\sum_{(x:A)} B(x), \mathsf{pr}_1\Big) : (A \to \mathcal{U}) \to \Big(\textstyle\sum_{(X:\mathcal{U})} X \to A\Big).$$

The fiber of this map at (X, f) is by univalence and function extensionality equivalent to the type

$$\textstyle\sum_{(B:A\to\mathcal{U})} \sum_{(e:(\sum_{(x:A)} B(x))\simeq X)} \mathsf{pr}_1 \sim f \circ e.$$

By Exercise 9.12 this type is equivalent to the type

$$\sum_{(B:A\to\mathcal{U})}\prod_{(a:A)}B(a)\simeq \mathsf{fib}_f(a),$$

and by 'type theoretic choice', which was established in Theorem 9.1.4, this type is equivalent to

$$\prod_{(a:A)}\sum_{(X:\mathcal{U})}X\simeq \mathsf{fib}_f(a).$$

We conclude that the fiber of φ at (X,f) is equivalent to the type $\mathsf{ess\_small}(f)$. However, since $f:X\to A$ is a map between small types it is essentially small. Moreover, since being essentially small is a proposition by Lemma 11.4.2, it follows that $\mathsf{fib}_\varphi((X,f))$ is contractible for every $f:X\to A$. In other words, φ is a contractible map, and therefore it is an equivalence. □

Remark 11.4.5. The inverse of the map

$$\varphi:(A\to\mathcal{U})\to\Big(\sum_{(X:\mathcal{U})}X\to A\Big).$$

constructed in Theorem 11.4.4 is the map $(X,f)\mapsto \mathsf{fib}_f$.

Theorem 11.4.6. *Let $f:A\to B$ be a map. Then there is an equivalence*

$$\mathsf{ess\_small}(f)\simeq\mathsf{is\_classified}(f),$$

where $\mathsf{is\_classified}(f)$ is the type of quadruples $(F,\tilde{F},H,p)$ consisting of maps $F:B\to\mathcal{U}$ and $\tilde{F}:A\to\sum_{(X:\mathcal{U})}X$, a homotopy $H:F\circ f\sim\mathsf{pr}_1\circ\tilde{F}$, such that the commuting square

$$\begin{tikzcd} A \arrow[r, "\tilde{F}"] \arrow[d, "f"'] & \sum_{(X:\mathcal{U})}X \arrow[d, "\mathsf{pr}_1"] \\ B \arrow[r, "F"'] & \mathcal{U} \end{tikzcd}$$

is a pullback square, as witnessed by p[2]*. If f comes equipped with a term of type $\mathsf{is\_classified}(f)$, we also say that f is **classified** by the universal family.*

Proof. From Exercise 9.13 we obtain that the type of pairs $(\tilde{F},H)$ is equivalent to the type of fiberwise transformations

$$\prod_{(b:B)}\mathsf{fib}_f(b)\to F(b).$$

[2] The universal property of the pullback is not expressible by a type. However, we may take the type of $p:\mathsf{is\_equiv}(h)$, where $h:A\to B\times_{\mathcal{U}}\big(\sum_{(X:\mathcal{U})}X\big)$ is the map obtained by the universal property of the canonical pullback.

By Corollary 10.5.4 the square is a pullback square if and only if the induced map

$$\prod_{(b:B)} \mathsf{fib}_f(b) \to F(b)$$

is a fiberwise equivalence. Thus the data $(F, \tilde{F}, H, pb)$ is equivalent to the type of pairs (F, e) where e is a fiberwise equivalence from fib_f to F. By Theorem 9.1.4 the type of pairs (F, e) is equivalent to the type $\mathsf{ess\_small}(f)$. □

Remark 11.4.7. For any type A (not necessarily small), and any $B : A \to \mathcal{U}$, the square

$$\begin{tikzcd} \sum_{(x:A)} B(x) \arrow[r, "\lambda(x,y).\,(B(x),y)"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(X:\mathcal{U})} X \arrow[d, "\mathsf{pr}_1"] \\ A \arrow[r, "B"'] & \mathcal{U} \end{tikzcd}$$

is a pullback square. Therefore it follows that for any family $B : A \to \mathcal{U}$ of small types, the projection map $\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$ is an essentially small map. To see that the claim is a direct consequence of Lemma 10.5.1 we write the asserted square in its rudimentary form:

$$\begin{tikzcd} \sum_{(x:A)} \mathrm{El}(B(x)) \arrow[r, "\lambda(x,y).\,(B(x),y)"] \arrow[d, "\mathsf{pr}_1"'] & \sum_{(X:\mathcal{U})} \mathrm{El}(X) \arrow[d, "\mathsf{pr}_1"] \\ A \arrow[r, "B"'] & \mathcal{U}. \end{tikzcd}$$

In the following theorem we show that a type is small if and only if its diagonal is classified by $\mathcal{U}$.

Theorem 11.4.8. *Let A be a type. The following are equivalent:*

(i) A is locally small.

(ii) There are maps $I : A \times A \to \mathcal{U}$ and $\tilde{I} : A \to \sum_{(X:\mathcal{U})} X$, and a homotopy $H : I \circ \delta_A \sim \mathsf{pr}_1 \circ \tilde{I}$ such that the commuting square

$$\begin{tikzcd} A \arrow[r, "\tilde{I}"] \arrow[d, "\delta_A"'] & \sum_{(X:\mathcal{U})} X \arrow[d, "\mathsf{pr}_1"] \\ A \times A \arrow[r, "I"'] & \mathcal{U} \end{tikzcd}$$

is a pullback square.

Proof. In Exercise 8.1 we have established that the identity type $x = y$ is the fiber of δ_A at $(x, y) : A \times A$. Therefore it follows that A is locally small if and only if the diagonal δ_A is essentially small. Now the result follows from Theorem 11.4.6. □

Exercises

11.1 Show that for any $P : X \to \mathcal{U}$ and any $p : x = y$ in X, we have

$$\mathsf{equiv\_eq}(\mathsf{ap}_P\,(p)) \sim \mathsf{tr}_P(p).$$

11.2 (a) Use the univalence axiom to show that the type $\sum_{(A:\mathcal{U})} \mathsf{is\_contr}(A)$ of all contractible types in $\mathcal{U}$ is contractible.

(b) Use Corollaries 8.3.4 and 9.1.3 and Exercise 9.3 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

$$\mathcal{U}^{\leq k} :\equiv \textstyle\sum_{(X:\mathcal{U})} \mathsf{is\_trunc}_k(X)$$

is a $(k+1)$-type, for any $k \geq -2$.

(d) It follows that the universe of propositions $\mathcal{U}^{\leq -1}$ is a set. However, 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.

11.3 Use the univalence axiom to show that the type $\sum_{(P:\mathsf{Prop})} P$ is contractible.

11.4 Let A and B be small types.

(a) Construct an equivalence

$$(A \to (B \to \mathcal{U})) \simeq \Big(\textstyle\sum_{(S:\mathcal{U})} (S \to A) \times (S \to B)\Big)$$

(b) We say that a relation $R : A \to (B \to \mathcal{U})$ is **functional** if it comes equipped with a term of type

$$\mathsf{is\_function}(R) :\equiv \textstyle\prod_{(x:A)} \mathsf{is\_contr}\Big(\textstyle\sum_{(y:B)} R(x, y)\Big)$$

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

$$\left(\sum_{(R:A\to(B\to\mathcal{U}))}\mathsf{is\_function}(R)\right)\simeq(A\to B)$$

(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

$$\left(\sum_{(R:A\to(B\to\mathcal{U}))}\mathsf{is\_function}(R)\times\mathsf{is\_function}(R^{\mathsf{op}})\right)\simeq(A\simeq B).$$

11.5 (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.

11.6 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 10.2 is classified by the universal fibration.

Lecture 12

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*. Moreover, 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.

12.1 The universal property of the circle

The *circle* is defined as a higher inductive type $\mathbb{S}^1$ that comes equipped with

$$\begin{aligned} &\mathsf{base} : \mathbb{S}^1 \\ &\mathsf{loop} : \mathsf{base} = \mathsf{base}. \end{aligned}$$

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 $f : \prod_{(t:\mathbb{S}^1)} P(t)$ 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 4.4.2 we also obtain for any dependent function $f : \prod_{(t:\mathbb{S}^1)} P(t)$ 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 12.1.1. Let P be a type family over the circle. The **dependent action on generators** is the map

$$\mathsf{dgen}_{\mathbb{S}^1} : \Big(\prod_{(t:\mathbb{S}^1)} P(t)\Big) \to \Big(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\Big) \tag{12.1}$$

given by $\mathsf{dgen}_{\mathbb{S}^1}(f) :\equiv (f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop}))$.

We now give the full specification of the circle.

Definition 12.1.2. The **circle** is a type $\mathbb{S}^1$ that comes equipped with

$$\begin{aligned} &\mathsf{base} : \mathbb{S}^1 \\ &\mathsf{loop} : \mathsf{base} = \mathsf{base}, \end{aligned}$$

and satisfies the **induction principle of the circle**, which provides for each type family P over $\mathbb{S}^1$ a map

$$\mathsf{ind}_{\mathbb{S}^1} : \Big(\sum_{(y:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, y) = y\Big) \to \Big(\prod_{(t:\mathbb{S}^1)} P(t)\Big),$$

and a homotopy witnessing that $\mathsf{ind}_{\mathbb{S}^1}$ is a section of $\mathsf{dgen}_{\mathbb{S}^1}$

$$\mathsf{dgen}_{\mathbb{S}^1} \circ \mathsf{ind}_{\mathbb{S}^1} \sim \mathsf{id}$$

for the computation rule.

Remark 12.1.3. The induction principle of the circle provides us with a dependent function $f : \prod_{(t:\mathbb{S}^1)} P(t)$ equipped with an identification

$$(f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop})) = (x, p),$$

for any $x : P(\mathsf{base})$ and $p : \mathsf{tr}_P(\mathsf{loop}, x) = x$. By Theorem 5.3.1 the identification $(f(\mathsf{base}), \mathsf{apd}_f(\mathsf{loop})) = (x, p)$ is equivalently described as a pair of identifications

$$\begin{aligned} &\alpha : f(\mathsf{base}) = x \\ &\beta : \mathsf{tr}(\alpha, \mathsf{apd}_f(\mathsf{loop})) = p. \end{aligned}$$

Here, the transport is taken with respect to the family $x \mapsto \mathsf{tr}_P(\mathsf{loop}, x) = x$.

The identity type $\mathsf{tr}(\alpha, \mathsf{apd}_f(\mathsf{loop})) = p$ is equivalent to the type

$$\mathsf{apd}_f(\mathsf{loop}) \cdot \alpha = \mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha) \cdot p.$$

Indeed, such an equivalence can be constructed by path induction, because types reduce to the type $\mathsf{apd}_f(\mathsf{loop}) = p$ when $\alpha \equiv \mathsf{refl}_{f(x)}$. Therefore we obtain from the computation rule of the circle an identification $\alpha : f(\mathsf{base}) = x$, and an identification

$$\beta' : \mathsf{apd}_f(\mathsf{loop}) \cdot \alpha = \mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha) \cdot p$$

witnessing that the square

$$\begin{tikzcd}
\mathsf{tr}_P(\mathsf{loop}, f(\mathsf{base})) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{loop})}(\alpha)"] \arrow[d, equal, "\mathsf{apd}_f(\mathsf{loop})"'] & \mathsf{tr}_P(\mathsf{loop}, x) \arrow[d, equal, "p"] \\
f(\mathsf{base}) \arrow[r, equal, "\alpha"'] & x
\end{tikzcd}$$

commutes.

For the remainder of this section we establish the **universal property** of the circle. The proof of Theorem 12.1.4 requires Lemmas 12.1.5 and 12.1.6, which we state after we encounter their application.

Theorem 12.1.4. *For each type X, the* ***action on generators***

$$\mathsf{gen}_{\mathbb{S}^1} : (\mathbb{S}^1 \to X) \to \textstyle\sum_{(x:X)} x = x$$

given by $f \mapsto (f(\mathsf{base}), \mathsf{ap}_f(\mathsf{loop}))$ is an equivalence.

Proof. Let $x : X$ and let $p : x = x$. By Exercise 4.3 we have an identification

$$\mathsf{tr\_triv}(\mathsf{loop}, x) : \mathsf{tr}_{W_{\mathbb{S}^1}X}(\mathsf{loop}, x) = x,$$

from which we obtain a fiberwise equivalence

$$\varphi : \textstyle\prod_{(x:X)} (x = x) \to (\mathsf{tr}_{W_{\mathbb{S}^1}X}(\mathsf{loop}, x) = x)$$

given by $p \mapsto \mathsf{tr\_triv}(\mathsf{loop}, x) \cdot p$. Moreover, for any $f : A \to B$, and any $p : x = y$ there is an identification $\mathsf{tr\_triv}(p, f(x)) \cdot \mathsf{ap}_f(p) = \mathsf{apd}_f(p)$, so it follows that the triangle

$$\begin{tikzcd}
& (\mathbb{S}^1 \to X) \arrow[dl, "\mathsf{gen}_{\mathbb{S}^1}"'] \arrow[dr, "\mathsf{dgen}_{\mathbb{S}^1}"] & \\
\sum_{(x:X)} x = x \arrow[rr, "\simeq", "\mathsf{total}(\varphi)"'] & & \sum_{(x:X)} \mathsf{tr}_{W_{\mathbb{S}^1}X}(\mathsf{loop}, x) = x \arrow[ul, dotted, bend right, "\mathsf{ind}_{\mathbb{S}^1}"']
\end{tikzcd}$$

commutes, and the map $\mathsf{total}(\varphi)$ is a fiberwise equivalence by Theorem 7.1.3. Since the triangle commutes and $\mathsf{ind}_{\mathbb{S}^1}$ is a section of $\mathsf{dgen}_{\mathbb{S}^1}$, it follows that the composite

$$\mathsf{rec}_{\mathbb{S}^1} :\equiv \mathsf{ind}_{\mathbb{S}^1} \circ \mathsf{total}(\varphi)$$

is a section of $\mathsf{gen}_{\mathbb{S}^1}$. Therefore it remains to show that $\mathsf{rec}_{\mathbb{S}^1}$ is also a retraction of $\mathsf{gen}_{\mathbb{S}^1}$, i.e. we have to show that for every $f : \mathbb{S}^1 \to X$ there is an identification

$$\mathsf{rec}_{\mathbb{S}^1}(\mathsf{gen}_{\mathbb{S}^1}(f)) = f.$$

In Lemma 12.1.5 below we establish that

$$(\mathsf{gen}_{\mathbb{S}^1}(\mathsf{rec}_{\mathbb{S}^1}(\mathsf{gen}_{\mathbb{S}^1}(f))) = \mathsf{gen}_{\mathbb{S}^1}(f)) \to (\mathsf{rec}_{\mathbb{S}^1}(\mathsf{gen}_{\mathbb{S}^1}(f)) = f).$$

We get an identification $\mathsf{gen}_{\mathbb{S}^1}(\mathsf{rec}_{\mathbb{S}^1}(\mathsf{gen}_{\mathbb{S}^1}(f))) = \mathsf{gen}_{\mathbb{S}^1}(f)$ from the fact that $\mathsf{rec}_{\mathbb{S}^1}$ is a section of $\mathsf{gen}_{\mathbb{S}^1}$. □

Lemma 12.1.5. *Let $f, g : \mathbb{S}^1 \to X$ be two dependent functions. Then there is a map*

$$(\mathsf{gen}_{\mathbb{S}^1}(f) = \mathsf{gen}_{\mathbb{S}^1}(g)) \to (f = g)$$

Proof. Let $p : \mathsf{gen}_{\mathbb{S}^1}(f) = \mathsf{gen}_{\mathbb{S}^1}(g)$. By function extensionality, it suffices to show that $f \sim g$. However, since $f \sim g$ is just the type $\prod_{(t:\mathbb{S}^1)} f(t) = g(t)$, we can construct such a homotopy by $\mathbb{S}^1$-induction. Thus, it suffices to construct a term of type

$$\textstyle\sum_{(p:f(\mathsf{base})=g(\mathsf{base}))} \mathsf{tr}_{E_{f,g}}(\mathsf{loop}, p) = p,$$

where $E_{f,g}$ is the family over $\mathbb{S}^1$ given by $t \mapsto f(t) = g(t)$.

We claim that it suffices to construct for each $p : f(\mathsf{base}) = g(\mathsf{base})$ an equivalence

$$\Big(\mathsf{tr}_{E_{f,g}}(\mathsf{loop}, p) = p\Big) \simeq \Big(\mathsf{tr}_L(p, \mathsf{ap}_f(\mathsf{loop})) = \mathsf{ap}_g(\mathsf{loop})\Big),$$

where L is the family over X given by $x \mapsto x = x$. To see that this suffices, we note that such a fiberwise equivalence induces an equivalence on total spaces, and the total space

$$\textstyle\sum_{(p:f(\mathsf{base})=g(\mathsf{base}))} \mathsf{tr}_L(p, \mathsf{ap}_f(\mathsf{loop})) = \mathsf{ap}_g(\mathsf{loop}),$$

and is equivalent to $\mathsf{gen}(f) = \mathsf{gen}(g)$, of which we have assumed a term.

The asserted fiberwise equivalence that we need for this proof to go through requires a sufficient generalization so that it can be constructed by path induction, so it is established separately in Lemma 12.1.6 below. □

With the following lemma we complete the proof of the universal property of the circle.

Lemma 12.1.6.

(i) *Let $f, g : A \to B$, and let $E_{f,g}$ be the family over A given by*

$$E_{f,g}(x) :\equiv f(x) = g(x).$$

Then for any $p : x = x'$ in A there is an equivalence

$$(\mathsf{tr}_{E_{f,g}}(p, q) = q') \simeq (\mathsf{ap}_f\,(p) \cdot q' = q \cdot \mathsf{ap}_g\,(p)).$$

for any $q : f(x) = g(x)$ and $q' : f(x') = g(x')$. In other words, there is an identification $\mathsf{tr}_{E_{f,g}}(p, q) = q'$ if and only if the square

$$\begin{tikzcd} f(x) \arrow[r, equal, "q"] \arrow[d, equal, "\mathsf{ap}_f(p)"'] & g(x) \arrow[d, equal, "\mathsf{ap}_g(p)"] \\ f(x') \arrow[r, equal, "q'"'] & g(x') \end{tikzcd}$$

commutes.

(ii) *Let L be the family over B given by $L(y) :\equiv y = y$, and let $q : y = y'$ be an identification in B. Then there is an equivalence*

$$(\mathsf{tr}_L(q, p) = p') \simeq (q \cdot p' = p \cdot q).$$

for any $p : y = y$ and $p' : y' = y'$. In other words, there is an identification $\mathsf{tr}_L(q, p) = p'$ if and only if the square

$$\begin{tikzcd} y \arrow[r, equal, "p"] \arrow[d, equal, "q"'] & y \arrow[d, equal, "q"] \\ y' \arrow[r, equal, "p'"'] & y' \end{tikzcd}$$

commutes.

(iii) *Let $f, g : A \to B$, let $p : x = x$ be a loop in A, and let $q : f(x) = g(x)$. Then there is an equivalence*

$$(\mathsf{tr}_{E_{f,g}}(p, q) = q) \simeq (\mathsf{tr}_L(q, \mathsf{ap}_f\,(p)) = \mathsf{ap}_g\,(p)).$$

Proof. The first claim follows by path induction on p, and the second claim follows by path induction on q. The third claim follows by combining the first two, since the types on both sides are equivalent to the type

$$\mathsf{ap}_f\,(p) \cdot q = q \cdot \mathsf{ap}_g\,(p)$$

of witnesses that the square

$$\begin{tikzcd} f(x) \arrow[r, equal, "q"] \arrow[d, equal, "\mathsf{ap}_f(p)"'] & g(x) \arrow[d, equal, "\mathsf{ap}_g(p)"] \\ f(x) \arrow[r, equal, "q"'] & g(x) \end{tikzcd}$$

commutes. □

12.2 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 $\mathbb{R} \to \mathbb{S}^1$ that winds the real line around the circle. In homotopy type theory there is no analogue of such a construction.

The type of small families over $\mathbb{S}^1$ is just the function type $\mathbb{S}^1 \to \mathcal{U}$, 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 $\mathbb{S}^1$ 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

$$\mathsf{eq\_equiv}_{X,X} : (X \simeq X) \to (X = X)$$

it suffices to provide an equivalence $X \simeq X$.

Definition 12.2.1. Consider a type X and every equivalence $e : X \simeq X$. We will construct a dependent type $\mathcal{D}(X, e) : \mathbb{S}^1 \to \mathcal{U}$ 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

$$\begin{aligned} (x_{\mathcal{D}})_X &= x & (e(x)_{\mathcal{D}}) &= \mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop}, x_{\mathcal{D}}) \\ (d_X)_{\mathcal{D}} &= d & (\mathsf{tr}_{\mathcal{D}(X,e)}(d))_X &= e(d_X) \end{aligned}$$

hold.

The type $\sum_{(X:\mathcal{U})} X \simeq X$ is also called the type of **descent data** for the circle.

Construction. By Exercise 11.1 we have an identification

$$\mathsf{equiv\_eq}(\mathsf{ap}_P(\mathsf{loop})) = \mathsf{tr}_P(\mathsf{loop})$$

for each dependent type $P : \mathbb{S}^1 \to \mathcal{U}$. Therefore we see that the triangle

$$\begin{tikzcd} & (\mathbb{S}^1 \to \mathcal{U}) \arrow[dl, "\mathsf{gen}_{\mathbb{S}^1}"'] \arrow[dr, "\mathsf{desc}_{\mathbb{S}^1}"] & \\ \sum_{(X:\mathcal{U})} X = X \arrow[rr, "\mathsf{total}(\lambda X.\, \mathsf{equiv\_eq}_{X,X})"'] & & \sum_{(X:\mathcal{U})} X \simeq X \end{tikzcd}$$

commutes, where the map $\mathsf{desc}_{\mathbb{S}^1}$ 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 7.1.3. Now it follows by the 3-for-2 property that $\mathsf{desc}_{\mathbb{S}^1}$ is an equivalence, since $\mathsf{gen}_{\mathbb{S}^1}$ is an equivalence by Theorem 12.1.4. This means that for every type X and every $e : X \simeq X$ there is a type family $\mathcal{D}(X,e) : \mathbb{S}^1 \to \mathcal{U}$ 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 $\mathsf{equiv\_eq}(p) : \mathcal{D}(X,e,\mathsf{base}) \simeq X$, for which the square

$$\begin{tikzcd} \mathcal{D}(X,e,\mathsf{base}) \arrow[r, "\mathsf{equiv\_eq}(p)"] \arrow[d, "\mathsf{tr}_{\mathcal{D}(X,e)}(\mathsf{loop})"'] & X \arrow[d, "e"] \\ \mathcal{D}(X,e,\mathsf{base}) \arrow[r, "\mathsf{equiv\_eq}(p)"'] & X \end{tikzcd}$$

commutes. □

Recall from Exercise 5.7 that the successor function $\mathsf{succ} : \mathbb{Z} \to \mathbb{Z}$ is an equivalence. Its inverse is the predecessor function defined in Exercise 3.11.

Definition 12.2.2. The **fundamental cover** of the circle is the dependent type $\mathcal{E}_{\mathbb{S}^1} :\equiv \mathcal{D}(\mathbb{Z}, \mathsf{succ}) : \mathbb{S}^1 \to \mathcal{U}$.

The picture of the fundamental cover is that of a helix over the circle.

Lemma 12.2.3. *For any $k : \mathbb{Z}$, there is an identification*

$$s_k : (\mathsf{base}, k_{\mathcal{E}}) = (\mathsf{base}, \mathsf{succ}(k)_{\mathcal{E}})$$

in the total space $\sum_{(t:\mathbb{S}^1)} \mathcal{E}(t)$.

Proof. By Theorem 5.3.1 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 $\mathcal{E}$. □

Our goal in this section is to show that the total space of the fundamental cover is contractible. We will use the following elimination principle for the integers.

Lemma 12.2.4. *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 3.3.2. 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 12.2.5. 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 the map

$$k \mapsto p^k : \mathbb{Z} \to (x = x).$$

Theorem 12.2.6. *The total space* $\sum_{(t:\mathbb{S}^1)} \mathcal{E}(t)$ *of the fundamental cover of* $\mathbb{S}^1$ *is contractible.*

Proof. We show that the total space satisfies singleton induction (i.e. we apply Theorem 6.1.1). Let P be a family over the total space of the fundamental cover, and let $p_0 : P(\mathsf{base}, 0_{\mathcal{E}})$. Our goal is to construct a term of type

$$\prod_{(t:\mathbb{S}^1)} \prod_{(x:\mathcal{E}(t))} P(t, x).$$

We do this by induction. For the base case we must construct a term of type

$$\prod_{(k:\mathbb{Z})} P(\mathsf{base}, k_{\mathcal{E}}).$$

Since we have the identifications $s_k : (\mathsf{base}, k_{\mathcal{E}}) = (\mathsf{base}, \mathsf{succ}(k)_{\mathcal{E}})$, we have the equivalences

$$\mathsf{tr}_P(s_k) : P(\mathsf{base}, k_{\mathcal{E}}) \simeq P(\mathsf{base}, \mathsf{succ}(k)_{\mathcal{E}})$$

for each $k : \mathbb{Z}$. Thus we obtain a dependent function $f : \prod_{(x:\mathcal{E}(\mathsf{base}))} P(\mathsf{base}, x)$ satisfying $f(0_{\mathcal{E}}) = p_0$ and $f(\mathsf{succ}(k)_{\mathcal{E}}) = \mathsf{tr}_P(s_k, f(k_{\mathcal{E}}))$, for each $k : \mathbb{Z}$.

For the loop case we must show that

$$\mathsf{tr}_Q(\mathsf{loop}, f) = f,$$

where Q is the family over $\mathbb{S}^1$ given by $Q(t) :\equiv \prod_{(x:\mathcal{E}(t))} P(t,x)$. By function extensionality it suffices to construct a homotopy, and the transport along loop in Q computes as

$$\mathsf{tr}_Q(\mathsf{loop}, f)(k_{\mathcal{E}}) = \mathsf{tr}_P(s_k, f(\mathsf{succ}^{-1}(k)_{\mathcal{E}})).$$

Therefore the following computation completes the proof:

$$\begin{aligned} \mathsf{tr}_Q(\mathsf{loop}, f)(k_{\mathcal{E}}) &= \mathsf{tr}_P(s_k, f(\mathsf{succ}^{-1}(k)_{\mathcal{E}})) \\ &= f(\mathsf{succ}(\mathsf{succ}^{-1}(k))_{\mathcal{E}}) \\ &= f(k_{\mathcal{E}}). \end{aligned}$$ □

Corollary 12.2.7. *We have a fiber sequence*

$$\mathbb{Z} \hookrightarrow \mathbf{1} \twoheadrightarrow \mathbb{S}^1.$$

In other words: the loop space $\Omega(\mathbb{S}^1)$ of the circle is equivalent to $\mathbb{Z}$.

Proof. This follows from Theorem 12.2.6 by an application of Theorem 7.2.1. □

Corollary 12.2.8. *The circle is a 1-type and it is not a 0-type.*

Exercises

12.1 Show that

$$\begin{tikzcd} X^{\mathbb{S}^1} \arrow[r, "-\circ\mathsf{const}_{\mathsf{base}}"] \arrow[d, "-\circ\mathsf{const}_{\mathsf{base}}"'] & X^{\mathbf{1}} \arrow[d, "-\circ\mathsf{const}_{\star}"] \\ X^{\mathbf{1}} \arrow[r, "-\circ\mathsf{const}_{\star}"'] & X^{\mathbf{2}} \end{tikzcd}$$

is a pullback square for each type X.

12.2 In this exercise we establish the *dependent universal property* of the circle, analogous to the proof of Theorem 12.1.4.

(a) Let $f, g : \prod_{(x:A)} B(x)$, and let $E_{f,g}$ be the family over A given by

$$E_{f,g}(x) :\equiv f(x) = g(x).$$

Construct for any $p : x = x'$ in A an equivalence

$$(\mathsf{tr}_{E_{f,g}}(p,q) = q') \simeq (\mathsf{apd}_f(p) \cdot q' = \mathsf{ap}_{\mathsf{tr}_B(p)}(q) \cdot \mathsf{apd}_g(p)).$$

for any $q : f(x) = g(x)$ and $q' : f(x') = g(x')$.

(b) Let B be a family over A, and for $l : x =_A x$ let L_x be the family over $B(x)$ given by

$$L_x(y) :\equiv \mathsf{tr}_B(l, y) = y.$$

Furthermore, let $q : y = y'$ be an identification in $B(x)$. Construct an equivalence

$$(\mathsf{tr}_{L_x}(q, p) = p') \simeq (\mathsf{ap}_{\mathsf{tr}_B(l)}(q) \cdot p' = p \cdot q).$$

for any $p : \mathsf{tr}_B(l, y) = y$ and $p' : \mathsf{tr}_B(l, y') = y'$.

(c) Let $f, g : \prod_{(x:A)} B(x)$, let $p : x = x$ be a loop in A, and let $q : f(x) = g(x)$. Construct an equivalence

$$(\mathsf{tr}_{E_{f,g}}(p, q) = q) \simeq (\mathsf{tr}_{L_x}(q, \mathsf{apd}_f(p)) = \mathsf{apd}_g(p)).$$

(d) Show that for any $f, g : \prod_{(t:\mathbb{S}^1)} P(t)$ there is a function

$$\Big(\mathsf{dgen}_{\mathbb{S}^1}(f) = \mathsf{dgen}_{\mathbb{S}^1}(g)\Big) \to (f = g).$$

(e) Show that for any type family P over $\mathbb{S}^1$, the *dependent action on generators*

$$\Big(\prod_{(t:\mathbb{S}^1)} P(t)\Big) \to \sum_{(u:P(\mathsf{base}))} \mathsf{tr}_P(\mathsf{loop}, u) = u$$

is an equivalence.

12.3 Let $P : \mathbb{S}^1 \to \mathsf{Prop}$ be a family of propositions over the circle. Show that

$$P(\mathsf{base}) \to \prod_{(t:\mathbb{S}^1)} P(t).$$

In this sense the circle is *connected.*

12.4 Show that

$$\prod_{(x,y:\mathbb{S}^1)} \neg\neg(x = y).$$

12.5 Use the fundamental cover of the circle to show that

$$\neg\Big(\prod_{(t:\mathbb{S}^1)} \mathsf{base} = t\Big).$$

12.6 Show that for any type X and any $x : X$, the map

$$\mathsf{rec}_{\mathbb{S}^1}(x, \mathsf{refl}_x) : \mathbb{S}^1 \to X$$

is homotopic to the constant map const_x.

12.7 (a) Show that for every $x : X$, we have an equivalence

$$\Big(\textstyle\sum_{(f:\mathbb{S}^1\to X)} f(\mathsf{base}) = x\Big) \simeq (x = x)$$

(b) Show that for every $t : \mathbb{S}^1$, we have an equivalence

$$\Big(\textstyle\sum_{(f:\mathbb{S}^1\to \mathbb{S}^1)} f(\mathsf{base}) = t\Big) \simeq \mathbb{Z}$$

The base point preserving map $f : \mathbb{S}^1 \to \mathbb{S}^1$ 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 $t : \mathbb{S}^1$, we have an equivalence

$$\Big(\textstyle\sum_{(e:\mathbb{S}^1\simeq \mathbb{S}^1)} e(\mathsf{base}) = t\Big) \simeq \mathbf{2}$$

12.8 The **(twisted) double cover** of the circle is defined as the type family $\mathcal{T} :\equiv \mathcal{D}(\mathbf{2}, \mathsf{neg}) : \mathbb{S}^1 \to \mathcal{U}$, where $\mathsf{neg} : \mathbf{2} \simeq \mathbf{2}$ is the negation equivalence of Exercise 5.6.

(a) Show that $\neg(\prod_{(t:\mathbb{S}^1)} \mathcal{T}(t))$.

(b) Construct an equivalence $e : \mathbb{S}^1 \simeq \sum_{(t:\mathbb{S}^1)} \mathcal{T}(t)$ for which the triangle

$$\begin{tikzcd}
\mathbb{S}^1 \arrow[rr, "e"] \arrow[dr, "\mathsf{deg}(2)"'] & & \sum_{(t:\mathbb{S}^1)} \mathcal{T}(t) \arrow[dl, "\mathsf{pr}_1"] \\
& \mathbb{S}^1 &
\end{tikzcd}$$

commutes.

12.9 (a) Show that a type X is a set if and only if the map

$$\lambda x.\, \lambda t.\, x : X \to (\mathbb{S}^1 \to X)$$

is an equivalence.

(b) Show that a type X is a set if and only if the map

$$\lambda f.\, f(\mathsf{base}) : (\mathbb{S}^1 \to X) \to X$$

is an equivalence.

12.10 Show that $(\mathbb{S}^1 \simeq \mathbb{S}^1) \simeq \mathbb{S}^1 + \mathbb{S}^1$. Conclude that a univalent universe containing a circle is not a 1-type.

12.11 Show that any retract of the circle is equivalent to the circle.

12.12 (a) Construct a fiberwise equivalence

$$\prod_{(t:\mathbb{S}^1)}\big((t = t) \simeq \mathbb{Z}\big).$$

(b) Use Exercise 12.9 to show that $(\mathsf{id}_{\mathbb{S}^1} \sim \mathsf{id}_{\mathbb{S}^1}) \simeq \mathbb{Z}$.

(c) Use Exercise 9.6 to show that

$$\mathsf{is\_invertible}(\mathsf{id}_{\mathbb{S}^1}) \simeq \mathbb{Z},$$

and conclude that $\mathsf{is\_invertible}(\mathsf{id}_{\mathbb{S}^1}) \not\simeq \mathsf{is\_equiv}(\mathsf{id}_{\mathbb{S}^1})$.

Lecture 13

Homotopy pushouts

We can use higher inductive types to attach cells to types. For example, when we are given a type A, and we have a map $f : \mathbb{S}^1 \to A$ describing a circle in A. Then we can form a new type A' in which we attach a disc by 'gluing' the boundary of the disc to the circle in A. Using higher inductive types, this process of attaching a disc works as follows:

(i) First we add all the points of A to A', i.e. A' comes equipped with a map

$$i : A \to A'$$

(ii) Next, we add a new point, which is to be thought of as the center of the disc that we're attaching. In other words, A' comes equipped with

$$\mathrm{pt} : A'$$

(iii) Finally, for each point x on the circle we add a path from the center of the disc to $i(f(x))$. In other words, A' comes equipped with a path constructor

$$r : \textstyle\prod_{(x:\mathbb{S}^1)} \mathrm{pt} = i(f(x)).$$

Moreover, since we're only attaching a disc to A along f, we suppose that A' satisfies an induction principle with respect to the constructors i, pt, and r.

The process of attaching a disc to a type A along a map $f : \mathbb{S}^1 \to A$ can be generalized, so that we will also be able to attach cells of different shapes to a type. This generalization is called homotopy pushouts. Homotopy pushouts are dual to homotopy pullbacks. However, unlike pullbacks we will *assume* that pushouts exist by postulating rules for higher inductive types. For the purpose of this course, the only higher inductive types that we add to our type theory are the pushouts. Some of the more exotic higher inductive types, including the Cauchy real numbers, are described in [2].

13.1 Pushouts as higher inductive types

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

$$\begin{tikzcd} A & S \arrow[l, "f"'] \arrow[r, "g"] & B. \end{tikzcd}$$

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 $f(x)$ to $g(x)$ for any $x : S$. Indeed, an equivalence between the type of all spans and the type of relations from A to B is established in Exercise 13.1.

Given a span $\mathcal{S}$ from A to B, we form the higher inductive type $A \sqcup^{\mathcal{S}} B$. It comes equipped with the following constructors

$$\begin{aligned} \mathsf{inl} &: A \to A \sqcup^{\mathcal{S}} B \\ \mathsf{inr} &: B \to A \sqcup^{\mathcal{S}} B \\ \mathsf{glue} &: \textstyle\prod_{(x:S)} \mathsf{inl}(f(x)) = \mathsf{inr}(g(x)) \end{aligned}$$

and we require that it satisfies an induction principle and computation rules.

To see what the induction principle has to be, consider first a dependent function $s : \prod_{(x:A \sqcup^{\mathcal{S}} B)} P(x)$. When we evaluate this function at the constructors, we obtain

$$\begin{aligned} s \circ \mathsf{inl} &: \textstyle\prod_{(a:A)} P(\mathsf{inl}(a)) \\ s \circ \mathsf{inr} &: \textstyle\prod_{(b:B)} P(\mathsf{inr}(b)) \\ \mathsf{apd}_s \circ \mathsf{glue} &: \textstyle\prod_{(x:S)} \mathsf{tr}_P(\mathsf{glue}(x), s(f(x))) = s(g(x)). \end{aligned}$$

Definition 13.1.1. Consider a span $\mathcal{S} \equiv (S, f, g)$ from A to B, and let P be a family over $A \sqcup^{\mathcal{S}} B$. The **dependent action on generators** is defined to be the map

$$\begin{aligned} \mathsf{dgen}_{\mathcal{S}}^{P} : \Big(\textstyle\prod_{(x:A \sqcup^{\mathcal{S}} B)} P(x)\Big) \to \Big(&\textstyle\sum_{(f':\prod_{(a:A)} P(\mathsf{inl}(a)))} \sum_{(g':\prod_{(b:B)} P(\mathsf{inr}(b)))} \\ &\textstyle\prod_{(x:S)} \mathsf{tr}_P(\mathsf{glue}(x), f'(f(x))) = g'(g(x))\Big). \end{aligned}$$

given by $s \mapsto (s \circ \mathsf{inl}, s \circ \mathsf{inr}, \mathsf{apd}_s \circ \mathsf{glue})$.

We can now fully specify homotopy pushouts.

Definition 13.1.2. Given a span $\mathcal{S} \equiv (S, f, g)$, the **(homotopy) pushout** $A \sqcup^{\mathcal{S}} B$ of $\mathcal{S}$ is defined to be the higher inductive type equipped with

$$\begin{aligned} \mathsf{inl} &: A \to A \sqcup^{\mathcal{S}} B \\ \mathsf{inr} &: B \to A \sqcup^{\mathcal{S}} B \\ \mathsf{glue} &: \textstyle\prod_{(x:S)} \mathsf{inl}(f(x)) = \mathsf{inr}(g(x)), \end{aligned}$$

satisfying the **induction principle** for pushouts, which asserts that for each type family P over $A \sqcup^{\mathcal{S}} B$ the map $\mathsf{dgen}_{\mathcal{S}}^{P}$ has a section.

Remark 13.1.3. The induction principle of the pushout $A \sqcup^{\mathcal{S}} B$ provides us with a dependent function

$$\mathsf{ind}_{\mathcal{S}}(f', g', G) : \textstyle\prod_{(x:A \sqcup^{\mathcal{S}} B)} P(x),$$

for every

$$\begin{aligned} f' &: \textstyle\prod_{(a:A)} P(\mathsf{inl}(a)) \\ g' &: \textstyle\prod_{(b:B)} P(\mathsf{inr}(b)) \\ G &: \textstyle\prod_{(x:S)} \mathsf{tr}_P(\mathsf{glue}(x), f'(f(x))) = g'(g(x)) \end{aligned}$$

Moreover, the function $\mathsf{ind}_{\mathcal{S}}(f', g', G)$ comes equipped with an identification

$$\mathsf{dgen}_{\mathcal{S}}(\mathsf{ind}_{\mathcal{S}}(f', g', G)) = (f', g', G).$$

Writing $s :\equiv \mathsf{ind}_{\mathcal{S}}(f', g', G)$, we see that such an identification between triples is equivalently described by a triple (H, K, L) consisting of

$$\begin{aligned} H &: s \circ \mathsf{inl} \sim f' \\ K &: s \circ \mathsf{inr} \sim g' \end{aligned}$$

and a homotopy L witnessing that the square

$$\begin{tikzcd} \mathsf{tr}_P(\mathsf{glue}(x), s(\mathsf{inl}(f(x)))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{tr}_P(\mathsf{glue}(x))}(H(x))"] \arrow[d, equal, "\mathsf{apd}_s(\mathsf{glue}(x))"'] & \mathsf{tr}_P(\mathsf{glue}(x), f'(f(x))) \arrow[d, equal, "G(x)"] \\ s(\mathsf{inr}(g(x))) \arrow[r, equal, "K(x)"'] & g'(g(x)) \end{tikzcd}$$

commutes, for every $x : S$. These are the **computation rules** for pushouts.

13.2 Examples of pushouts

Many interesting types can be defined as homotopy pushouts.

Definition 13.2.1. Let X be a type. We define the **suspension** ΣX of X to be the pushout of the span

$$\begin{tikzcd} X \arrow[r] \arrow[d] & \mathbf{1} \arrow[d, "\mathsf{inr}"] \\ \mathbf{1} \arrow[r, "\mathsf{inl}"'] & \Sigma X \end{tikzcd}$$

Definition 13.2.2. We define the n**-sphere** $\mathbb{S}^n$ for any $n : \mathbb{N}$ by induction on n, by taking

$$\mathbb{S}^0 :\equiv \mathbf{2}$$
$$\mathbb{S}^{n+1} :\equiv \Sigma\mathbb{S}^n.$$

Definition 13.2.3. 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 13.2.4. The suspension ΣX of X is the cofiber of the map $X \to \mathbf{1}$.

Definition 13.2.5. 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 13.2.6. 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 13.2.7. 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

$$\begin{tikzcd}
I \arrow[r, "\lambda i.\,(i,a_0(i))"] \arrow[d] & \sum_{(i:I)} A_i \arrow[d] \\
\mathbf{1} \arrow[r] & \bigvee_{(i:I)} A_i.
\end{tikzcd}$$

13.3 The universal property of pushouts

Definition 13.3.1. 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.

Definition 13.3.2. 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

$$\mathsf{cocone\_map}(i, j, H) : (X \to Y) \to \mathsf{cocone}(Y)$$

by $f \mapsto (f \circ i, f \circ j, f \cdot H)$.

Definition 13.3.3. 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

$$\mathsf{cocone\_map}(i, j, H) : (X \to Y) \to \mathsf{cocone}(Y)$$

is an equivalence for any type Y. Sometimes pushout squares are also called **cocartesian squares**.

Lemma 13.3.4. *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 $\pi_3' :\equiv \lambda(i, j, H).\, \mathsf{eq\_htpy}(H)$, is a pullback square.

Proof. The gap map $\mathsf{cocone}_{\mathcal{S}}(X) \to X^A \times_{X^S} X^B$ is the function

$$\lambda(i, j, H).\, (i, j, \mathsf{eq\_htpy}(H)).$$

This is an equivalence by Theorem 7.1.3, since it is the induced map on total spaces of the fiberwise equivalence $\mathsf{eq\_htpy}$. Therefore, the square is a pullback square by Theorem 10.2.6. □

In the following theorem we establish an alternative characterization of the universal property of pushouts.

Theorem 13.3.5. *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

$$\lambda h.\, \mathsf{eq\_htpy}(h \cdot H)$$

is a pullback square, for every type T.

Proof. It is straightforward to verify that the triangle

$$\begin{tikzcd}
 & T^X \arrow[dl, "\mathsf{cocone\_map}(i,j,H)"'] \arrow[dr, "\mathsf{gap}(-\circ i, -\circ j, \mathsf{eq\_htpy}(-\cdot H))"] & \\
\mathsf{cocone}(T) \arrow[rr, "\mathsf{gap}(i,j,\mathsf{eq\_htpy}(H))"'] & & T^A \times_{T^S} T^B
\end{tikzcd}$$

commutes. Since the bottom map is an equivalence by Lemma 13.3.4, it follows that if either one of the remaining maps is an equivalence, so is the other. The claim now follows by Theorem 10.2.6. □

Example 13.3.6. By Exercise 12.1 and the second characterization of pushouts in Theorem 13.3.5 it follows that the circle is a pushout

$$\begin{tikzcd}
\mathbf{2} \arrow[r] \arrow[d] & \mathbf{1} \arrow[d] \\
\mathbf{1} \arrow[r] & \mathbb{S}^1.
\end{tikzcd}$$

In other words, $\mathbb{S}^1 \simeq \mathbf{\Sigma 2}$.

Theorem 13.3.7. *Consider a span $\mathcal{S} \equiv (S, f, g)$ from A to B. Then the square*

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "\mathsf{inr}"] \\
A \arrow[r, "\mathsf{inl}"'] & A \sqcup^{\mathcal{S}} B
\end{tikzcd}$$

is a pushout square.

Proof. Let X be a type. Our goal is to show that the map

$$\mathsf{cocone\_map}(\mathsf{inl}, \mathsf{inr}, \mathsf{glue}) : (A \sqcup^{\mathcal{S}} B \to X) \to \mathsf{cocone}_{\mathcal{S}}(X)$$

is an equivalence. For notational breveity we will just write $\mathsf{gen}_{\mathcal{S}}$ for $\mathsf{cocone\_map}_{\mathcal{S}}(\mathsf{inl}, \mathsf{inr}, \mathsf{glue})$, because $\mathsf{cocone\_map}_{\mathcal{S}}(\mathsf{inl}, \mathsf{inr}, \mathsf{glue})$ is just the action on generators.

We first note that by Exercise 4.3 there is a commuting triangle

$$\begin{tikzcd}
 & X^{A \sqcup^{\mathcal{S}} B} \arrow[dl, "\mathsf{gen}_{\mathcal{S}}"'] \arrow[dr, "\mathsf{dgen}_{\mathcal{S}}"] & \\
\mathsf{cocone}_{\mathcal{S}}(X) \arrow[rr, "\simeq"] & & \mathsf{cocone}'_{\mathcal{S}}(X)
\end{tikzcd}$$

where we write

$$\mathsf{cocone}'_{\mathcal{S}}(X) : \Big(\textstyle\sum_{(f':A\to X)}\sum_{(g':A\to X)} \\ \prod_{(x:S)} \mathsf{tr}_{W_{A\sqcup^{\mathcal{S}}B}(X)}(\mathsf{glue}(x), f'(f(x))) = g'(g(x))\Big).$$

By the induction principle for $A \sqcup^{\mathcal{S}} B$ we have a section $\mathsf{ind}_{\mathcal{S}}$ of $\mathsf{dgen}_{\mathcal{S}}$. Thus we obtain a section $\mathsf{rec}_{\mathcal{S}}$ of $\mathsf{gen}_{\mathcal{S}}$. Our goal is now to show that $\mathsf{rec}_{\mathcal{S}}$ is also a retraction of $\mathsf{gen}_{\mathcal{S}}$. We establish in Lemma 13.3.8 that

$$(\mathsf{gen}_{\mathcal{S}}(\mathsf{rec}_{\mathcal{S}}(\mathsf{gen}_{\mathcal{S}}(h))) = \mathsf{gen}_{\mathcal{S}}(h)) \to (\mathsf{rec}_{\mathcal{S}}(\mathsf{gen}_{\mathcal{S}}(h)) = h)$$

Then we obtain that $\mathsf{rec}_{\mathcal{S}}$ is a retraction of $\mathsf{gen}_{\mathcal{S}}$ by using this implication and the fact that $\mathsf{rec}_{\mathcal{S}}$ is a section of $\mathsf{gen}_{\mathsf{S}}$. □

Lemma 13.3.8. *Let $h, h' : A \sqcup^{\mathcal{S}} B \to X$ be two functions. Then we have*

$$(\mathsf{gen}_{\mathcal{S}}(h) = \mathsf{gen}_{\mathcal{S}}(h')) \to (h = h').$$

Proof. Suppose we have $\mathsf{gen}_{\mathcal{S}}(h) = \mathsf{gen}_{\mathcal{S}}(h')$. This type of equalities between triples is equivalent to the type of triples (K, L, M) consisting of

$$K : h \circ \mathsf{inl} \sim h' \circ \mathsf{inl}$$
$$L : h \circ \mathsf{inr} \sim h' \circ \mathsf{inr},$$

and a homotopy M witnessing that the square

$$\begin{tikzcd}
h \circ \mathsf{inl} \circ f \arrow[r, "K \cdot f"] \arrow[d, "h \cdot \mathsf{glue}"'] & h' \circ \mathsf{inl} \circ f \arrow[d, "h' \cdot \mathsf{glue}"] \\
h \circ \mathsf{inr} \circ f \arrow[r, "L \cdot g"'] & h' \circ \mathsf{inr} \circ g
\end{tikzcd}$$

of homotopies commutes. By function extensionality, our goal is equivalent to constructing a homotopy (i.e. a dependent function) of type

$$\prod_{(t:A \sqcup^{\mathcal{S}} B)} f(t) = g(t).$$

We will construct such a function by the induction principle for $A \sqcup^{\mathcal{S}} B$. Therefore it suffices to construct

$$\begin{aligned} K &: h \circ \mathsf{inl} \sim h' \circ \mathsf{inl} \\ L &: h \circ \mathsf{inr} \sim h' \circ \mathsf{inr} \\ M' &: \mathsf{tr}_{E_{h,h'}}(\mathsf{glue}, K) = L \end{aligned}$$

The type of M' is equivalent to the type of M, so we obtain the requested structure from our assumptions. □

As a basic application we establish the universal property of suspensions.

Corollary 13.3.9. *Let X and Y be types. Then the map*

$$(\Sigma X \to Y) \to \sum_{(y,y':Y)} X \to (y = y')$$

given by $f \mapsto (f(\mathsf{inl}(\star)), f(\mathsf{inr}(\star)), \mathsf{ap}_f\,(\mathsf{glue}(\mathord{-})))$ is an equivalence.

Proof. We have equivalences

$$\begin{aligned} (\Sigma X \to Y) &\simeq \sum_{(y,y':\mathbf{1} \to Y)} X \to (y(\star) = y'(\star)) \\ &\simeq \sum_{(y,y':Y)} X \to (y = y'). \end{aligned}$$

□

13.4 The pasting property for pushouts

Theorem 13.4.1. *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 10.6.1 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 13.3.5. □

Lemma 13.4.2. *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.*

Exercises

13.1 Use Theorems 9.1.4 and 11.4.4 and Corollary 9.2.2 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.

13.2 Use Theorems 9.3.3 and 13.3.5 and Corollary 10.5.6 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 $X \simeq \mathbf{0} * X$.

(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.

13.3 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 $R :\equiv \mathsf{is\_contr}(P * Q)$ to show that $P * Q$ is again a proposition.

13.4 Let Q be a proposition, and let A be a type. Show that the following are equivalent:

(a) The map $(Q \to A) \to (\mathbf{0} \to A)$ is an equivalence.
(b) The type A^Q is contractible.
(c) There is a term of type $Q \to \mathsf{is\_contr}(A)$.
(d) The map $\mathsf{inr} : A \to Q * A$ is an equivalence.

13.5 Let P be a proposition. Show that ΣP is a set, with an equivalence

$$\Big(\mathsf{inl}(\star) = \mathsf{inr}(\star)\Big) \simeq P.$$

13.6 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 $\mathcal{S}$.

13.7 Use Exercise 10.6.b 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.

13.8 Use Exercise 10.5 to show that if

$$\begin{tikzcd}
S_1 \arrow[r] \arrow[d] & Y_1 \arrow[d] & S_2 \arrow[r] \arrow[d] & Y_2 \arrow[d] \\
X_1 \arrow[r] & Z_1 & X_2 \arrow[r] & Z_2
\end{tikzcd}$$

are pushout squares, then so is

$$\begin{tikzcd}
S_1 + S_2 \arrow[r] \arrow[d] & Y_1 + Y_2 \arrow[d] \\
X_1 + X_2 \arrow[r] & Z_1 + Z_2.
\end{tikzcd}$$

13.9 (a) Consider a span (S, f, g) from A to B. Use Exercise 10.4 to show that the square

$$\begin{tikzcd}
S + S \arrow[r, "{[\mathsf{id},\mathsf{id}]}"] \arrow[d, "f+g"'] & S \arrow[d, "\mathsf{inr} \circ g"] \\
A + B \arrow[r, "{[\mathsf{inl},\mathsf{inr}]}"'] & A \sqcup^{S} B
\end{tikzcd}$$

is again a pushout square.

(b) Show that $\Sigma X \simeq \mathbf{2} * X$.

13.10 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 Exercise 10.10 to show that $\mathsf{cofib}_{\mathsf{cofib}(h,H)} \simeq \mathsf{cofib}_h$.

13.11 Use Exercise 12.9 to show that for $n \geq 0$, X is an n-type if and only if the map

$$\lambda x. \mathsf{const}_x : X \to (\mathbb{S}^{n+1} \to X)$$

is an equivalence.

13.12 (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 $g : Y \to Z$.

13.13 Consider a commuting diagram of the form

$$\begin{tikzcd}
A_0 & B_0 \arrow[l] \arrow[r] & C_0 \\
A_1 \arrow[u] \arrow[d] & B_1 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & C_1 \arrow[u] \arrow[d] \\
A_2 & B_2 \arrow[l] \arrow[r] & C_2
\end{tikzcd}$$

with homotopies filling the (small) squares. Use Exercise 10.11 to construct an equivalence

$$\begin{aligned}(A_0 \sqcup^{B_0} C_0) &\sqcup^{(A_1 \sqcup^{B_1} C_1)} (A_2 \sqcup^{B_2} C_2) \\ &\simeq (A_0 \sqcup^{A_1} A_2) \sqcup^{(B_0 \sqcup^{B_1} B_2)} (C_0 \sqcup^{C_1} C_2).\end{aligned}$$

This is known as the **3-by-3 lemma** for pushouts.

13.14 (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

$$\Big(\bigvee_{(x:X)} \mathbf{2}\Big) \simeq X + 1.$$

(c) Construct an equivalence

$$\Sigma(\mathsf{Fin}(n+1)) \simeq \bigvee_{(i:\mathsf{Fin}(n))} \mathbb{S}^1.$$

13.15 Show that $\mathsf{Fin}(n+1) * \mathsf{Fin}(m+1) \simeq \bigvee_{(i:\mathsf{Fin}(n\cdot m))} \mathbb{S}^1$, for any $n, m : \mathbb{N}$.

Lecture 14

Descent

14.1 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

$$\begin{aligned}
P \circ i &: A \to \mathcal{U} \\
P \circ j &: B \to \mathcal{U} \\
\lambda x.\, \mathsf{tr}_P(H(x)) &: \textstyle\prod_{(x:S)} P(i(f(x))) \simeq P(j(g(x))).
\end{aligned}$$

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 14.1.1. 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

$$P_A : A \to \mathcal{U}$$

$$\begin{aligned} P_B &: B \to \mathcal{U} \\ P_S &: \textstyle\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x)). \end{aligned}$$

Definition 14.1.2. 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

$$\mathsf{desc\_fam}_{\mathcal{S}}(i, j, H) : (X \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{S})$$

by $P \mapsto (P \circ i, P \circ j, \lambda x.\, \mathsf{tr}_P(H(x)))$.

Theorem 14.1.3. *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

$$\mathsf{desc\_fam}_{\mathcal{S}}(i, j, H) : (X \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{S})$$

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

$$\begin{tikzcd} & \mathcal{U}^X \arrow[dl, "\mathsf{cocone\_map}_{\mathcal{S}}(i,j,H)"'] \arrow[dr, "\mathsf{desc\_fam}_{\mathcal{S}}(i,j,H)"] & \\ \mathsf{cocone}_{\mathcal{S}}(\mathcal{U}) \arrow[rr, dotted, "\simeq", "\varphi"'] & & \mathsf{Desc}(\mathcal{S}) \end{tikzcd}$$

commutes.

Since we have equivalences

$$\mathsf{equiv\_eq} : \Big(P_A(f(x)) = P_B(g(x))\Big) \simeq \Big(P_A(f(x)) \simeq P_B(g(x))\Big)$$

for all $x : S$, we obtain by Exercise 9.11 an equivalence on the dependent products

$$\Big(\textstyle\prod_{(x:S)} P_A(f(x)) = P_B(g(x))\Big) \to \Big(\textstyle\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x))\Big).$$

We define φ to be the induced map on total spaces. Explicitly, we have

$$\varphi :\equiv \lambda(P_A, P_B, K).\,(P_A, P_B, \lambda x.\,\mathsf{equiv\_eq}(K(x))).$$

Then φ is an equivalence by Theorem 7.1.3, and the triangle commutes by Exercise 11.1. □

Corollary 14.1.4. *Consider descent data (P_A, P_B, P_S) for a pushout square as in Theorem 14.1.3. Then the type of quadruples (P, e_A, e_B, e_S) consisting of a family $P : X \to \mathcal{U}$ equipped with fiberwise equivalences*

$$e_A : \textstyle\prod_{(a:A)} P_A(a) \simeq P(i(a))$$
$$e_B : \textstyle\prod_{(b:B)} P_B(a) \simeq P(j(b))$$

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 6.3.3. □

14.2 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

$$\begin{aligned}
&P_A : A \to \mathcal{U}\\
&P_B : B \to \mathcal{U}\\
&P_S : \textstyle\prod_{(x:S)} P_A(f(x)) \simeq P_B(g(x)),
\end{aligned}$$

and a family $P : X \to \mathcal{U}$ equipped with

$$\begin{aligned}
&e_A : \textstyle\prod_{(a:A)} P_A(a) \simeq P(i(a))\\
&e_B : \textstyle\prod_{(b:B)} P_B(a) \simeq P(j(b))
\end{aligned}$$

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 14.2.1. 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 $H' : i' \circ f' \sim j' \circ g'$.

Construction. We define

$$\begin{aligned}
f' &:\equiv \mathsf{total}_f(\lambda x.\, \mathsf{id}_{P_A(f(x))})\\
g' &:\equiv \mathsf{total}_g(e_S)\\
i' &:\equiv \mathsf{total}_i(e_A)\\
j' &:\equiv \mathsf{total}_j(e_B).
\end{aligned}$$

The remaining goal is to construct a homotopy $H' : i' \circ f' \sim j' \circ g'$. Thus, we have to show that

$$(i(f(x)), e_A(y)) = (j(g(x)), e_B(e_S(y)))$$

for any $x : S$ and $y : P_A(f(x))$. We have he identification

$$\mathsf{eq\_pair}(H(x), e_S(x,y)^{-1})$$

of this type. □

Definition 14.2.2. We will write $\mathcal{S}'$ for the span

$$\textstyle\sum_{(a:A)} P_A(a) \xleftarrow{f'} \sum_{(x:S)} P_A(f(x)) \xrightarrow{g'} \sum_{(b:B)} P_B(b).$$

Lemma 14.2.3 (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. We will show that the map

$$\mathsf{cocone\_map}_{\mathcal{S}'}(i', j', H') : \Big(\Big(\textstyle\sum_{(x:X)} P(x)\Big) \to Y\Big) \to \mathsf{cocone}_{\mathcal{S}'}(Y)$$

is an equivalence for any type Y. Let Y be a type. Note that the type $\mathsf{cocone}_{\mathcal{S}'}$ is equivalent to the type of triples (u, v, w) consisting of

$$\begin{aligned}
u &: \textstyle\prod_{(a:A)} P_A(a) \to Y \\
v &: \textstyle\prod_{(b:B)} P_B(b) \to Y \\
w &: \textstyle\prod_{(x:S)} \prod_{(y:P_A(f(x)))} u(f(x), y) = v(g(x), e_S(x,y)).
\end{aligned}$$

Now observe that there is an equivalence

$$\begin{aligned}
&\Big(\textstyle\prod_{(y:P_A(f(x)))} u(f(x), y) = v(g(x), e_S(x,y))\Big) \\
&\qquad \simeq \mathsf{tr}_{(t \mapsto P(t) \to Y)}(H(x), u'(f(x))) = v'(g(x))
\end{aligned}$$

for any u and v as above, and any $x : S$. By this equivalence we obtain a commuting square

$$\begin{tikzcd}
\Big(\Big(\sum_{(x:X)} P(x)\Big) \to Y\Big) \arrow[r, "\mathsf{ind}_\Sigma", "\simeq"'] \arrow[d, "{\mathsf{cocone\_map}_{\mathcal{S}'}(i',j',H')}"'] & \prod_{(x:X)} (P(x) \to Y) \arrow[d, "\mathsf{dgen}_{\mathcal{S}}"] \\
\mathsf{cocone}_{\mathcal{S}'}(Y) \arrow[r, "\simeq"] & \Psi
\end{tikzcd}$$

where Ψ is the type of triples (u', v', w') consisting of

$$\begin{aligned}
u' &: \textstyle\prod_{(a:A)} P(i(a)) \to Y \\
v' &: \textstyle\prod_{(b:B)} P(j(b)) \to Y \\
w' &: \textstyle\prod_{(x:S)} \mathsf{tr}_{(t \mapsto P(t) \to Y)}(H(x), u'(f(x))) = v'(g(x)),
\end{aligned}$$

Since the dependent action on generators $\mathsf{dgen}_{\mathcal{S}}$ is an equivalence it follows by the 3-for-2 property of equivalences that $\mathsf{cocone\_map}_{\mathcal{S}'}(i', j', H')$ is an equivalence, as desired. □

14.3 Commuting cubes

Definition 14.3.1. A **commuting cube**

$$\begin{tikzcd}
 & A_{111} \arrow[dl] \arrow[d] \arrow[dr] & \\
A_{110} \arrow[d] \arrow[dr] & A_{101} \arrow[dl, crossing over] \arrow[dr] & A_{011} \arrow[dl, crossing over] \arrow[d] \\
A_{100} \arrow[dr] & A_{010} \arrow[d] & A_{001} \arrow[dl] \\
 & A_{000}, &
\end{tikzcd}$$

consists of

(i) types

$$A_{111}, A_{110}, A_{101}, A_{011}, A_{100}, A_{010}, A_{001}, A_{000},$$

(ii) maps

$$\begin{aligned}
f_{11\check{1}} &: A_{111} \to A_{110} & \qquad f_{\check{1}01} &: A_{101} \to A_{001} \\
f_{1\check{1}1} &: A_{111} \to A_{101} & f_{01\check{1}} &: A_{011} \to A_{010} \\
f_{\check{1}11} &: A_{111} \to A_{011} & f_{0\check{1}1} &: A_{011} \to A_{001} \\
f_{1\check{1}0} &: A_{110} \to A_{100} & f_{\check{1}00} &: A_{100} \to A_{000} \\
f_{\check{1}10} &: A_{110} \to A_{010} & f_{0\check{1}0} &: A_{010} \to A_{000} \\
f_{10\check{1}} &: A_{101} \to A_{100} & f_{00\check{1}} &: A_{001} \to A_{000},
\end{aligned}$$

(iii) homotopies

$$\begin{aligned} H_{1\check{1}\check{1}} &: f_{1\check{1}0} \circ f_{11\check{1}} \sim f_{10\check{1}} \circ f_{1\check{1}1} & H_{0\check{1}\check{1}} &: f_{0\check{1}0} \circ f_{01\check{1}} \sim f_{00\check{1}} \circ f_{0\check{1}1} \\ H_{\check{1}1\check{1}} &: f_{\check{1}10} \circ f_{11\check{1}} \sim f_{01\check{1}} \circ f_{\check{1}11} & H_{\check{1}0\check{1}} &: f_{\check{1}00} \circ f_{10\check{1}} \sim f_{00\check{1}} \circ f_{\check{1}01} \\ H_{\check{1}\check{1}1} &: f_{\check{1}01} \circ f_{1\check{1}1} \sim f_{0\check{1}1} \circ f_{\check{1}11} & H_{\check{1}\check{1}0} &: f_{\check{1}00} \circ f_{1\check{1}0} \sim f_{0\check{1}0} \circ f_{\check{1}10}, \end{aligned}$$

(iv) and a homotopy

$$\begin{aligned} C : (f_{\check{1}00} \cdot H_{1\check{1}\check{1}}) \centerdot ((H_{\check{1}0\check{1}} \cdot f_{1\check{1}1}) \centerdot (f_{00\check{1}} \cdot H_{\check{1}\check{1}1})) \\ \sim (H_{\check{1}\check{1}0} \cdot f_{11\check{1}}) \centerdot ((f_{0\check{1}0} \cdot H_{\check{1}1\check{1}}) \centerdot (H_{0\check{1}\check{1}} \cdot f_{\check{1}11})) \end{aligned}$$

filling the cube.

Lemma 14.3.2. *Given a commuting cube as in Definition 14.3.1 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 14.3.3. *Consider a commuting cube*

$$\begin{tikzcd} & & A_{111} \arrow[dll] \arrow[d] \arrow[drr] & & \\ A_{110} \arrow[d] \arrow[drr] & & A_{101} \arrow[dll] \arrow[drr] & & A_{011} \arrow[dll] \arrow[d] \\ A_{100} \arrow[drr] & & A_{010} \arrow[d] & & A_{001} \arrow[dll] \\ & & A_{000}. & & \end{tikzcd}$$

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 14.3.4. 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**.

14.4 The descent property for pushouts

In the previous section there was a significant role for fiberwise equivalences, and we know by Theorem 10.5.2 and Corollary 10.5.4: fiberwise equivalences indicate the presence of pullbacks. In this section we reformulate the results of the previous section using pullbacks where we used fiberwise equivalences before, to obtain new and useful results. We begin by considering the type of descent data from the perspective of pullback squares.

Definition 14.4.1. 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 $p_f : \mathsf{is\_pullback}(h_S, h_A, F)$ and $p_g : \mathsf{is\_pullback}(h_S, h_B, G)$, 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 14.4.2. *There is an equivalence*

$$\mathsf{cart\_desc}_{\mathcal{S}} : \mathsf{Desc}(\mathcal{S}) \to \mathsf{Cart}(\mathcal{S}).$$

Proof. Note that by Theorem 10.5.7 it follows that the types of triples (f', F, p_f) and (g', G, p_g) are equivalent to the types of fiberwise equivalences

$$\textstyle\prod_{(x:S)} \mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_A}(f(x))$$

$$\prod_{(x:S)}\mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_B}(g(x))$$

respectively. Furthermore, by Theorem 11.4.4 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

$$\begin{aligned} Q &: S \to \mathcal{U} \\ P_A &: A \to \mathcal{U} \\ P_B &: B \to \mathcal{U} \\ \varphi &: \prod_{(x:S)} Q(x) \simeq P_A(f(x)) \\ P_S &: \prod_{(x:S)} Q(x) \simeq P_B(g(x)). \end{aligned}$$

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. □

Definition 14.4.3. We define an operation

$$\mathsf{cart\_map}_{\mathcal{S}} : \Big(\sum_{(X':\mathcal{U})} X' \to X\Big) \to \mathsf{Cart}(\mathcal{S}).$$

Construction. Let $X' : \mathcal{U}$ and $h_X : X' \to X$. Then we define the types

$$\begin{aligned} A' &:\equiv A \times_X X' \\ B' &:\equiv B \times_X X'. \end{aligned}$$

Next, we define a span $\mathcal{S}' :\equiv (S', f', g')$ from A' to B'. We take

$$\begin{aligned} S' &:\equiv S \times_A A' \\ f' &:\equiv \pi_2. \end{aligned}$$

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 summaze, 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 10.6.1. □

The following theorem is analogous to Theorem 14.1.3.

Theorem 14.4.4 (The descent theorem for pushouts)**.** *The operation* $\mathsf{cart\_map}_{\mathcal{S}}$ *is an equivalence*

$$\left(\sum\nolimits_{(X':\mathcal{U})} X' \to X\right) \simeq \mathsf{Cart}(\mathcal{S})$$

Proof. It suffices to show that the square

$$\begin{tikzcd}
X \to \mathcal{U} \arrow[r, "\mathsf{desc\_fam}_{\mathcal{S}}(i,j,H)"] \arrow[d, "\mathsf{map\_fam}_X"'] & \mathsf{Desc}(\mathcal{S}) \arrow[d, "\mathsf{cart\_desc}_{\mathcal{S}}"] \\
\sum_{(X':\mathcal{U})} X' \to X \arrow[r, "\mathsf{cart\_map}_{\mathcal{S}}"'] & \mathsf{Cart}(\mathcal{S})
\end{tikzcd}$$

commutes. To see that this suffices, note that the operation $\mathsf{map\_fam}_X$ is an equivalence by Theorem 11.4.4, the operation $\mathsf{desc\_fam}_{\mathcal{S}}(i,j,H)$ is an equivalence by Theorem 14.1.3, and the operation $\mathsf{cart\_desc}_{\mathcal{S}}$ is an equivalence by Lemma 14.4.2.

To see that the square commutes, note that the composite

$$\mathsf{cart\_map}_{\mathcal{S}} \circ \mathsf{map\_fam}_X$$

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

$$\mathsf{cart\_desc}_{\mathcal{S}} \circ \mathsf{desc\_fam}_X$$

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 10.5.1 □

Since $\mathsf{cart\_map}_{\mathcal{S}}$ is an equivalence it follows that its fibers are contractible. This is essentially the content of the following corollary.

Corollary 14.4.5. *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

$$\begin{aligned}
F &: f \circ h_S \sim h_A \circ f' \\
G &: g \circ h_S \sim h_B \circ g' \\
H &: i \circ f \sim j \circ g,
\end{aligned}$$

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

$$C : (i \cdot F) \centerdot ((I \cdot f') \centerdot (h_X \cdot H')) \sim (H \cdot h_S) \centerdot ((j \cdot G) \centerdot (J \cdot g'))$$

witnessing that the cube

$$\begin{tikzcd}
& S' \arrow[dl] \arrow[d] \arrow[dr] & \\
A' \arrow[d] \arrow[dr] & S \arrow[dl, crossing over] \arrow[dr, crossing over] & B' \arrow[dl] \arrow[d] \\
A \arrow[dr] & X' \arrow[d] & B \arrow[dl] \\
& X, &
\end{tikzcd}$$

commutes,

is contractible.

The following theorem should be compared to the flattening lemma, Lemma 14.2.3.

Theorem 14.4.6. *Consider a commuting cube*

$$\begin{tikzcd}
& S' \arrow[dl, "f'"'] \arrow[d, "h_S"] \arrow[dr, "g'"] & \\
A' \arrow[d, "h_A"'] \arrow[dr, "i'"'] & S \arrow[dl, crossing over, "f"'] \arrow[dr, crossing over, "g"] & B' \arrow[dl, "j'"] \arrow[d, "h_B"] \\
A \arrow[dr, "i"'] & X' \arrow[d, "h_X"] & B \arrow[dl, "j"] \\
& X. &
\end{tikzcd}$$

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 Corollary 10.5.4 we have fiberwise equivalences

$$\begin{aligned}
F &: \textstyle\prod_{(x:S)}\mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_A}(f(x)) \\
G &: \textstyle\prod_{(x:S)}\mathsf{fib}_{h_S}(x) \simeq \mathsf{fib}_{h_B}(g(x)) \\
I &: \textstyle\prod_{(a:A)}\mathsf{fib}_{h_A}(a) \simeq \mathsf{fib}_{h_X}(i(a)) \\
J &: \textstyle\prod_{(b:B)}\mathsf{fib}_{h_B}(b) \simeq \mathsf{fib}_{h_X}(j(b)).
\end{aligned}$$

Moreover, since the cube commutes we obtain a fiberwise homotopy

$$K : \textstyle\prod_{(x:S)}I(f(x)) \circ F(x) \sim J(g(x)) \circ G(x).$$

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 13.2 that the left square is a pushout square. Moreover, the right square is a pushout by Lemma 14.2.3. Therefore it follows by Theorem 13.4.1 that the outer rectangle is a pushout square.

Now consider the commuting cube

$$\begin{tikzcd}
 & & \sum_{(s:S)} \mathsf{fib}_{h_S}(s) \arrow[dll] \arrow[d] \arrow[drr] & & \\
\sum_{(a:A)} \mathsf{fib}_{h_A}(a) \arrow[d] \arrow[drr] & & S' \arrow[dll, crossing over] \arrow[drr, crossing over] & & \sum_{(b:B)} \mathsf{fib}_{h_B}(b) \arrow[dll] \arrow[d] \\
A' \arrow[drr] & & \sum_{(x:X)} \mathsf{fib}_{h_X}(x) \arrow[d] & & B' \arrow[dll] \\
 & & X'. & &
\end{tikzcd}$$

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 13.4.1 that the bottom square is a pushout. □

Theorem 14.4.7. *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

$$\begin{tikzcd}
 & & S' \arrow[dll] \arrow[d] \arrow[drr] & & \\
A' \arrow[d] \arrow[drr] & & S \arrow[dll] \arrow[drr] & & B' \arrow[dll] \arrow[d] \\
A \arrow[drr] & & (A \sqcup^S B) \times_X X' \arrow[d] & & B \arrow[dll] \\
 & & A \sqcup^S B & &
\end{tikzcd}$$

are pullback squares by Theorem 10.6.1. □

Exercises

14.1 Use the characterization of the circle as a pushout given in Example 13.3.6 to show that the square

$$\begin{tikzcd}
\mathbb{S}^1 + \mathbb{S}^1 \arrow[r, "{[\mathsf{id},\mathsf{id}]}"] \arrow[d, "{[\mathsf{id},\mathsf{id}]}"'] & \mathbb{S}^1 \arrow[d, "{\lambda t.\,(t,\mathsf{base})}"] \\
\mathbb{S}^1 \arrow[r, "{\lambda t.\,(t,\mathsf{base})}"'] & \mathbb{S}^1 \times \mathbb{S}^1
\end{tikzcd}$$

is a pushout square.

14.2 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

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "f"'] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\mathsf{id}_B"] & \\
A \arrow[r, "\mathsf{inl}"] \arrow[urr, phantom, very near end, "\ulcorner"] \arrow[drr, bend right, "\mathsf{id}_B"'] & B \sqcup^A B \arrow[dr, dotted, "\nabla_f"] & \\
 & & B
\end{tikzcd}$$

Show that $\mathsf{fib}_{\nabla_f}(b) \simeq \Sigma(\mathsf{fib}_f(b))$ for any $b : B$.

14.3 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

$$\begin{tikzcd}
A \times_X B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "g"] & \\
A \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "f"'] & A *_X B \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dotted, "f*g"'] & \\
& & X
\end{tikzcd}$$

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 $x : X$.

14.4 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 $d : D$.

14.5 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:

$$\begin{tikzcd}
\mathbf{1} \arrow[r] \arrow[d] & B \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\lambda b.\, (a_0, b)"] & \\
A \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "\lambda a.\, (a, b_0)"'] & A \vee B \arrow[dr, dotted, "\mathsf{wedge\_in}_{A,B}"'] & \\
& & A \times B
\end{tikzcd}$$

Show that the fiber of the wedge inclusion $A \vee B \to A \times B$ is equivalent to $\Omega(B) * \Omega(A)$.

14.6 Let $f : X \vee X \to X$ be the map defined by the universal property of the wedge as indicated in the diagram

$$\begin{tikzcd}
\mathbf{1} \arrow[r, "x_0"] \arrow[d, "x_0"'] \arrow[dr, phantom, "\ulcorner", very near end] & X \arrow[d, "\mathsf{inr}"] \arrow[ddr, bend left, "\mathsf{id}_X"] & \\
X \arrow[r, "\mathsf{inl}"] \arrow[drr, bend right, "\mathsf{id}_X"'] & X \vee X \arrow[dr, dotted, "f"] & \\
 & & X.
\end{tikzcd}$$

Show that $\mathsf{fib}_f(x_0) \simeq \Sigma\Omega(X)$.

Lecture 15

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.

15.1 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 15.1.1. An **(increasing) type sequence** $\mathcal{A}$ consists of

$$\begin{aligned} &A : \mathbb{N} \to \mathcal{U} \\ &f : \textstyle\prod_{(n:\mathbb{N})} A_n \to A_{n+1}. \end{aligned}$$

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 15.1.2. (i) A **(sequential) cocone** on a type sequence $\mathcal{A}$ with vertex B consists of

$$h : \textstyle\prod_{(n:\mathbb{N})} A_n \to B$$

$$H : \prod_{(n:\mathbb{N})} f_n \sim f_{n+1} \circ H_n.$$

We write $\mathsf{cocone}(B)$ for the type of cones with vertex X.

(ii) Given a cone (h, H) with vertex B on a type sequence $\mathcal{A}$ we define the map

$$\mathsf{cocone\_map}(h, H) : (B \to C) \to \mathsf{cocone}(B)$$

given by $f \mapsto (f \circ h, \lambda n.\, \lambda x.\, \mathsf{ap}_f(H_n(x)))$.

(iii) We say that a cone (h, H) with vertex B is **colimiting** if $\mathsf{cocone\_map}(h, H)$ is an equivalence for any type C.

Theorem 15.1.3. *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.

15.2 The construction of sequential colimits

We construct sequential colimits using pushouts.

Definition 15.2.1. 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 15.2.2. The type A_∞ comes equipped with a cocone structure consisting of

$$\begin{aligned} \mathsf{seq\_in} &: \textstyle\prod_{(n:\mathbb{N})} A_n \to A_\infty \\ \mathsf{seq\_glue} &: \textstyle\prod_{(n:\mathbb{N})}\prod_{(x:A_n)} \mathsf{in}_n(x) = \mathsf{in}_{n+1}(f_n(x)). \end{aligned}$$

Construction. We define

$$\begin{aligned} \mathsf{seq\_in}(n,x) &:\equiv \mathsf{inr}(n,x) \\ \mathsf{seq\_glue}(n,x) &:\equiv \mathsf{glue}(\mathsf{inl}(n,x))^{-1} \cdot \mathsf{glue}(\mathsf{inr}(n,x)). \end{aligned}$$

□

Theorem 15.2.3. *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.

15.3 Descent for sequential colimits

Definition 15.3.1. The type of **descent data** on a type sequence $\mathcal{A} \equiv (A,f)$ is defined to be

$$\mathsf{Desc}(\mathcal{A}) :\equiv \textstyle\sum_{(B:\prod_{(n:\mathbb{N})} A_n \to \mathcal{U})}\prod_{(n:\mathbb{N})}\prod_{(x:A_n)} B_n(x) \simeq B_{n+1}(f_n(x)).$$

Definition 15.3.2. We define a map

$$\mathsf{desc\_fam} : (A_\infty \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{A})$$

by $B \mapsto (\lambda n.\, \lambda x.\, B(\mathsf{seq\_in}(n,x)), \lambda n.\, \lambda x.\, \mathsf{tr}_B(\mathsf{seq\_glue}(n,x)))$.

Theorem 15.3.3. *The map*

$$\mathsf{desc\_fam} : (A_\infty \to \mathcal{U}) \to \mathsf{Desc}(\mathcal{A})$$

is an equivalence.

Definition 15.3.4. 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

$$\begin{aligned} \mathsf{cart}(\mathcal{A}, \mathcal{B}) :\equiv & \textstyle\sum_{(h:\prod_{(n:\mathbb{N})} A_n \to B_n)} \\ & \textstyle\sum_{(H:\prod_{(n:\mathbb{N})} g_n \circ h_n \sim h_{n+1} \circ f_n)} \prod_{(n:\mathbb{N})} \mathsf{is\_pullback}(h_n, f_n, H_n), \end{aligned}$$

and we write

$$\mathsf{Cart}(\mathcal{B}) :\equiv \textstyle\sum_{(\mathcal{A}:\mathsf{Seq})} \mathsf{cart}(\mathcal{A}, \mathcal{B}).$$

Definition 15.3.5. We define a map

$$\mathsf{cart\_map}(\mathcal{B}) : \left(\textstyle\sum_{(X':\mathcal{U})} X' \to X\right) \to \mathsf{Cart}(\mathcal{B}).$$

which associates to any morphism $h : X' \to X$ a cartesian transformation of type sequences into $\mathcal{B}$.

Theorem 15.3.6. *The operation* $\mathsf{cart\_map}(\mathcal{B})$ *is an equivalence.*

15.4 The flattening lemma for sequential colimits

The flattening lemma for sequential colimits essentially states that sequential colimits commute with Σ.

Lemma 15.4.1. *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*

$$e : \textstyle\prod_{(n:\mathbb{N})} B_n(x) \simeq P(\mathsf{seq\_in}(n,x))$$

and homotopies $H_n(x)$ *witnessing that the square*

$$\begin{tikzcd} B_n(x) \arrow[r, "g_n(x)"] \arrow[d, "e_n(x)"'] & B_{n+1}(f_n(x)) \arrow[d, "e_{n+1}(f_n(x))"] \\ P(\mathsf{seq\_in}(n,x)) \arrow[r, "\mathsf{tr}_P(\mathsf{seq\_glue}(n,x))"'] & P(\mathsf{seq\_in}(n+1,f_n(x))) \end{tikzcd}$$

commutes. Then $\sum_{(t:A_\infty)} P(t)$ *satisfies the universal property of the sequential colimit of the type sequence*

$$\textstyle\sum_{(x:A_0)} B_0(x) \xrightarrow{\mathsf{total}_{f_0}(g_0)} \sum_{(x:A_1)} B_1(x) \xrightarrow{\mathsf{total}_{f_1}(g_1)} \sum_{(x:A_2)} B_2(x) \xrightarrow{\mathsf{total}_{f_2}(g_2)} \cdots .$$

In the following theorem we rephrase the flattening lemma in using cartesian transformations of type sequences.

Theorem 15.4.2. *Consider a commuting diagram of the form*

$$\begin{tikzcd} A_0 \arrow[r] \arrow[drr] \arrow[dd] & A_1 \arrow[rr] \arrow[dr] \arrow[dd] & & A_2 \arrow[r] \arrow[dl] \arrow[dd] & \cdots \arrow[dll] \\ & & X \arrow[dd] & & \\ B_0 \arrow[r] \arrow[drr] & B_1 \arrow[rr] \arrow[dr] & & B_2 \arrow[r] \arrow[dl] & \cdots \arrow[dll] \\ & & Y & & \end{tikzcd}$$

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 15.4.3. *Consider a commuting diagram of the form*

$$\begin{tikzcd}
A_0 \arrow[r] \arrow[rrd] \arrow[dd] & A_1 \arrow[rr] \arrow[rd] \arrow[dd] & & A_2 \arrow[r] \arrow[ld] \arrow[dd] & \cdots \arrow[lld] \\
& & X \arrow[dd] & & \\
B_0 \arrow[r] \arrow[rrd] & B_1 \arrow[rr] \arrow[rd] & & B_2 \arrow[r] \arrow[ld] & \cdots \arrow[lld] \\
& & Y & &
\end{tikzcd}$$

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.

Exercises

15.1 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

$$A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \cdots .$$

15.2 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.

15.3 Define the ∞-sphere $\mathbb{S}^\infty$ as the sequential colimit of

$$\mathbb{S}^0 \xrightarrow{f_0} \mathbb{S}^1 \xrightarrow{f_1} \mathbb{S}^2 \xrightarrow{f_2} \cdots$$

where $f_0 : \mathbb{S}^0 \to \mathbb{S}^1$ is defined by $f_0(0_{\mathbf{2}}) \equiv \mathsf{inl}(\star)$ and $f_0(1_{\mathbf{2}}) \equiv \mathsf{inr}(\star)$, and $f_{n+1} : \mathbb{S}^{n+1} \to \mathbb{S}^{n+2}$ is defined as $\Sigma(f_n)$. Use Exercise 15.2 to show that $\mathbb{S}^\infty$ is contractible.

15.4 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

$$\textstyle\prod_{(x,y:A_n)} f_n(x) = f_n(y)$$

Show that A_∞ is a mere proposition.

Lecture 16

The homotopy image of a map

16.1 The universal property of the image of a map

Definition 16.1.1. Let $f : A \to X$ and $g : B \to X$ be maps. We define

$$\hom_X(f,g) :\equiv \textstyle\sum_{(h:A\to B)} f \sim g \circ h.$$

Remark 16.1.2. In other words, a term $(h,H) : \hom_X(f,g)$ 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. Recall from Exercise 9.12 that the type $\hom_X(f,g)$ is equivalent to the type

$$\textstyle\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_g(x).$$

Lemma 16.1.3. *For any $f : A \to X$ and any embedding $m : B \to X$, the type $\hom_X(f,m)$ is a proposition.*

Proof. Since propositions are closed under equivalences by Lemma 8.1.5, it suffices to show that the type

$$\textstyle\prod_{(x:X)} \mathsf{fib}_f(x) \to \mathsf{fib}_m(x),$$

is a proposition. Recall from Corollary 8.3.8 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, so it is a proposition by Theorem 9.1.2. □

Definition 16.1.4. Consider a commuting triangle

$$\begin{tikzcd} A \arrow[rr, "i"] \arrow[dr, swap, "f"] & & B \arrow[dl, "m"] \\ & X & \end{tikzcd}$$

with $I : f \sim m \circ i$, and where m is an embedding. We say that m has the **universal property of the image of** f if the map

$$(i, I)^* : \hom_X(m, m') \to \hom_X(f, m')$$

defined by $(i, I)^*(h, H) :\equiv (h \circ i, I \centerdot (i \cdot H))$, is an equivalence for every embedding $m' : B' \to X$.

Remark 16.1.5. Since $\hom_X(f, m)$ is a proposition for every $f : A \to X$ and every embedding $m : B \to X$, it follows by Exercise 9.1 that the universal property of the image of f is equivalent to the property that the implication

$$\hom_X(f, m') \to \hom_X(m, m')$$

holds for every embedding $m' : B' \to X$.

The homotopy image can be used in many important constructions. In this lecture we discuss two applications: the propositional truncation, and set quotients.

16.2 The propositional truncation

Note that embeddings into the unit type are just propositions. To see this, note that

$$\begin{aligned} \textstyle\sum_{(A:\mathcal{U})} \sum_{(f:A \to \mathbf{1})} \mathsf{is\_emb}(f) &\simeq \textstyle\sum_{(A:\mathcal{U})} \mathsf{is\_emb}(\mathsf{const}_\star) \\ &\simeq \textstyle\sum_{(A:\mathcal{U})} \prod_{(x:\mathbf{1})} \mathsf{is\_prop}(\mathsf{fib}_{\mathsf{const}_\star}(x)) \\ &\simeq \textstyle\sum_{(A:\mathcal{U})} \mathsf{is\_prop}(\mathsf{fib}_{\mathsf{const}_\star}(\star)) \\ &\simeq \textstyle\sum_{(A:\mathcal{U})} \mathsf{is\_prop}(A). \end{aligned}$$

Therefore, the universal property of the image of the map $A \to \mathbf{1}$ is a proposition P satisfying the universal property of the propositional truncation:

Definition 16.2.1. Let A be a type, and let P be a proposition that comes equipped with a map $\eta : A \to P$. We say that $\eta : A \to P$ satisfies the **universal property of propositional truncation** if for every proposition Q, the precomposition map

$$- \circ \eta : (P \to Q) \to (A \to Q)$$

is an equivalence.

Theorem 16.2.2. *Consider a commuting triangle*

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

with $I : f \sim m \circ i$, and where m is an embedding. The following are equivalent:

(i) m satisfies the universal property of the image of f.

(ii) for each $x : X$, the proposition $\mathsf{fib}_m(x)$ satisfies the universal property of the propositional truncation of $\mathsf{fib}_f(x)$.

Note that, given a family of propositions P over a type A, the type $\sum_{(a:A)} P(a)$ isn't necessarily a proposition. Instead, we think of $\sum_{(a:A)} P(a)$ of the *subtype* of A containing the terms that satisfies P. Using the propositional truncation we can assert that there *exists* a term in A that satisfies P without requiring one to construct it.

Definition 16.2.3. Let $P : A \to \mathsf{Prop}$ be a family of propositions over a type A. Then we define

$$\exists_{(a:A)} P(a) :\equiv \left\| \textstyle\sum_{(a:A)} P(a) \right\|.$$

Similarly, we can define the disjuction of two propositions P and Q to be the *proposition* $\|P + Q\|$, which clearly satisfies the universal property of disjunction[1]. In Table 16.1 we give an overview of the logical connectives on propositions.

[1] Alternatively, we have shown in Exercise 13.3 that the join $P * Q$ also is a proposition that satisfies the universal property of disjunction.

Table 16.1: Logic in type theory

Logical connective	*Interpretation in HoTT*
$\top$	$\mathbf{1}$
$\bot$	$\mathbf{0}$
$P \wedge Q$	$P \times Q$
$P \vee Q$	$\lVert P + Q \rVert$
$P \to Q$	$P \to Q$
$P \leftrightarrow Q$	$P \simeq Q$
$\neg P$	$P \to \mathbf{0}$
$\forall x.P(x)$	$\prod_{(x:A)} P(x)$
$\exists x.P(x)$	$\lVert \sum_{(x:A)} P(x) \rVert$
$\exists ! x.P(x)$	$\mathsf{is\_contr}(\sum_{(x:A)} P(x))$

16.3 Constructing the propositional truncation

Although technically it is not necessary to construct the propositional truncation before constructing the image of a map, we do so because the construction is simpler in this special case, and yet contains most of the essential ideas.

Lemma 16.3.1. *Suppose $f : A \to P$, where A is any type, and P is a proposition. Then the map*

$$(A * B \to P) \to (B \to P)$$

given by $h \mapsto h \circ \mathsf{inr}$ is an equivalence, for any type B.

Proof. Since both types are propositions by Theorem 9.1.2 it suffices to construct a map

$$(B \to P) \to (A * B \to P).$$

Let $g : B \to P$. Then the square

$$\begin{tikzcd} A \times B \arrow[r, "\mathsf{pr}_2"] \arrow[d, "\mathsf{pr}_1"'] & B \arrow[d, "g"] \\ A \arrow[r, "f"'] & P \end{tikzcd}$$

commutes since P is a proposition. Therefore we obtain a map $A * B \to P$ by the universal property of the join. □

The idea of the construction of the propositional truncation is that if we are given a map $f : A \to P$, where P is a proposition, then it extends uniquely along $\mathsf{inr} : A \to A * A$ to a map $A * A \to P$. This extension again extends uniquely along $\mathsf{inr} : A * A \to A * (A * A)$ to a map $A * (A * A) \to P$ and so on, resulting in a diagram of the form

$$\begin{tikzcd}
A \arrow[r, "\mathsf{inr}"] \arrow[dr] & A * A \arrow[r, "\mathsf{inr}"] \arrow[d, dotted] & A * (A * A) \arrow[r, "\mathsf{inr}"] \arrow[dl, dotted] & \cdots \arrow[dll, dotted, bend left] \\
 & P & &
\end{tikzcd}$$

Definition 16.3.2. The **join powers** A^{*n} of a type X are defined by

$$\begin{aligned}
A^{*0} &:\equiv \mathbf{0} \\
A^{*1} &:\equiv A \\
A^{*(n+1)} &:\equiv A * A^{*n}.
\end{aligned}$$

Furthermore, we define $A^{*\infty}$ to be the sequential colimit of the type sequence

$$A^{*0} \longrightarrow A^{*1} \xrightarrow{\mathsf{inr}} A^{*2} \xrightarrow{\mathsf{inr}} \cdots .$$

Our goal is now to show that $A^{*\infty}$ is a proposition and satisfies the universal property of the propositional truncation.

Lemma 16.3.3. *Consider a type sequence*

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots$$

with sequential colimit A_∞, and let P be a proposition. Then the map

$$\mathsf{seq\_in}^* : (A_\infty \to P) \to \left(\prod_{(n:\mathbb{N})} A_n \to P\right)$$

given by $h \mapsto \lambda n.\,(h \circ \mathsf{seq\_in}_n)$ is an equivalence.

Proof. By the universal property of sequential colimits established in Theorem 15.1.3 we obtain that $\mathsf{cocone\_map}$ is an equivalence. Note that we have a commuting triangle

$$\begin{tikzcd}
 & P^{A_\infty} \arrow[dl, "\mathsf{cocone\_map}"'] \arrow[dr, "\mathsf{seq\_in}^*"] & \\
\mathsf{cocone}) \arrow[rr, "\mathsf{pr}_1"'] & & \left(\prod_{(n:\mathbb{N})} A_n \to P\right).
\end{tikzcd}$$

Note that for any $g : \prod_{(n:\mathbb{N})} A_n \to P$ the type

$$\prod_{(n:\mathbb{N})} g_n \sim g_{n+1} \circ f_n$$

is a product of contractible types, since P is a proposition. Therefore it is contractible by Theorem 9.1.1, and it follows by Exercise 7.5 that the projection is an equivalence. We conclude by the 3-for-2 property of equivalences (Exercise 5.5) that $\mathsf{seq\_in}^*$ is an equivalence. □

Lemma 16.3.4. *Let A be a type, and let P be a proposition. Then the function*

$$- \circ \mathsf{seq\_in}_0 : (A^{*\infty} \to P) \to (A \to P)$$

is an equivalence.

Proof. We have the commuting triangle

$$\begin{tikzcd}
& P^{A^{*\infty}} \arrow[dl, "\mathsf{seq\_in}^*"'] \arrow[dr, "-\circ\mathsf{seq\_in}_0"] & \\
\left(\prod_{(n:\mathbb{N})} A^{*n} \to P\right) \arrow[rr, "\lambda h.\, h_0"'] & & P^A.
\end{tikzcd}$$

Therefore it suffices to show that the bottom map is an equivalence. Since this is a map between propositions, it suffices to construct a map in the converse direction. Let $f : A \to P$. We will construct a term of type

$$\prod_{(n:\mathbb{N})} A^{*n} \to P$$

by induction on $n : \mathbb{N}$. The base case is trivial. Given a map $g : A^{*n} \to P$, we obtain a map $g : A^{*(n+1)} \to P$ by Lemma 16.3.1. □

Lemma 16.3.5. *The type $A^{*\infty}$ is a proposition for any type A.*

Proof. By Corollary 8.1.4 it suffices to show that $A^{*\infty} \to \mathsf{is\_contr}(A^{*\infty})$, and by Lemma 16.3.4 it suffices to show that

$$A \to \mathsf{is\_contr}(A^{*\infty}),$$

because $\mathsf{is\_contr}(A^{*\infty})$ is a proposition by Exercise 9.2.

Let $x : A$. To see that $A^{*\infty}$ is contractible it suffices by Exercise 15.2 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. □

Theorem 16.3.6. *For any type A there is a type $\|A\|$ that comes equipped with a map $\eta : A \to \|A\|$, and satisfies the universal property of propositional truncation.*

Proof. Let A be a type. Then we define $\|A\| :\equiv A^{*\infty}$, and we define $\eta :\equiv \mathsf{seq\_in}_0 : A \to A^{*\infty}$. Then $\|A\|$ is a proposition by Theorem 16.4.6, and $\eta : A \to \|A\|$ satisfies the universal property of propositional truncation by Lemma 16.3.4. □

16.4 The construction of the image of a map

The image of a map $f : A \to X$ can be defined using the propositional truncation:

Definition 16.4.1. For any map $f : A \to X$ we define the **image** of f to be the type

$$\mathrm{im}(f) :\equiv \textstyle\sum_{(x:X)} \|\mathsf{fib}_f(x)\|$$

and we define the **image inclusion** to be the projection $\mathsf{pr}_1 : \mathrm{im}(f) \to X$.

However, the construction of the fiberwise join in Exercise 14.3 suggests that we can also define the image of f as the infinite join power $f^{*\infty}$, where we repeatedly take the fiberwise join of f with itself. The reasons for defining the image in this way are twofold: we will be able to use this construction to show that the set-quotients of a small type are small, and second, we many interesting types appear in this construction.

Lemma 16.4.2. *Consider a map $f : A \to X$, an embedding $m : U \to X$, and $h : \mathrm{hom}_X(f, m)$. Then the map*

$$\mathrm{hom}_X(f * g, m) \to \mathrm{hom}_X(g, m)$$

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

$$\begin{aligned}
\mathrm{hom}_X(f * g, m) &\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_{f*g}(x) \to \mathsf{fib}_m(x) \\
&\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_f(x) * \mathsf{fib}_g(x) \to \mathsf{fib}_m(x) \\
&\simeq \textstyle\prod_{(x:X)} \mathsf{fib}_g(x) \to \mathsf{fib}_m(x) \\
&\simeq \mathrm{hom}_X(g, m).
\end{aligned}$$

The first equivalence holds by Exercise 9.12; the second equivalence holds by Exercise 14.3, also using Theorem 9.3.3 and Exercise 9.5 where we established that that pre- and postcomposing by an equivalence is an equivalence; the third equivalence holds by Lemma 16.3.1 and Exercise 9.5; the last equivalence again holds by Exercise 9.12. □

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 16.4.3. Suppose $f : A \to X$ is a map. Then we define the **fiberwise join powers**

$$f^{*n} : A^{*n}_X X.$$

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 $(\mathbf{0}, \mathsf{ind}_{\mathbf{0}})$ and (A, f) of this type. For $n \geq 1$ we define

$$\begin{aligned}
A^{*(n+1)}_X &:\equiv A *_X A^{*n}_X \\
f^{*(n+1)} &:\equiv f * f^{*n}.
\end{aligned}$$

□

Definition 16.4.4. We define $A^{*\infty}_X$ to be the sequential colimit of the type sequence

$$\begin{tikzcd}
A^{*0}_X \arrow[r] & A^{*1}_X \arrow[r, "\mathsf{inr}"] & A^{*2}_X \arrow[r, "\mathsf{inr}"] & \cdots .
\end{tikzcd}$$

Since we have a cocone

$$\begin{tikzcd}
A^{*0}_X \arrow[r] \arrow[dr, "f^{*0}"'] & A^{*1}_X \arrow[r, "\mathsf{inr}"] \arrow[d, "f^{*1}"'] & A^{*2}_X \arrow[r, "\mathsf{inr}"] \arrow[dl, "f^{*2}"'] & \cdots \arrow[dll, bend left] \\
& X & &
\end{tikzcd}$$

we also obtain a map $f^{*\infty} : A^{*\infty}_X \to X$ by the universal property of $A^{*\infty}_X$.

Lemma 16.4.5. *Let $f : A \to X$ be a map, and let $m : U \to X$ be an embedding. Then the function*

$$- \circ \mathsf{seq\_in}_0 : \hom_X(f^{*\infty}, m) \to \hom_X(f, m)$$

is an equivalence.

Theorem 16.4.6. *For any map $f : A \to X$, the map $f^{*\infty} : A^{*\infty}_X \to X$ is an embedding that satisfies the universal property of the image inclusion of f.*

16.5 Surjective maps

Another application of the propositional truncation is the notion of surjective map.

Definition 16.5.1. A map $f : A \to B$ is said to be **surjective** if there is a term of type

$$\mathsf{is\_surj}(f) :\equiv \textstyle\prod_{(y:B)} \|\mathsf{fib}_f(b)\|.$$

Example 16.5.2. 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 $\mathbf{1} \to \mathbb{S}^n$ for any $n \geq 1$.

Theorem 16.5.3. *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 m satisfies the universal property of the image of f if and only if $i : A \to B$ is surjective.

Theorem 16.5.4. *Let $f : A \to B$ be a map. The following are equivalent:*

(i) f is an equivalence.

(ii) f is both surjective and an embedding.

Exercises

16.1 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.

16.2 For any $B : A \to \mathcal{U}$, construct an equivalence

$$\left(\exists_{(a:A)}\|B(a)\|\right) \simeq \left\|\sum_{(a:A)} B(a)\right\|$$

16.3 Let $P_0 \longrightarrow P_1 \longrightarrow P_2 \longrightarrow \cdots$ be a sequence of propositions. Show that

$$\mathsf{colim}_n(P_n) \simeq \exists_{(n:\mathbb{N})} P_n.$$

16.4 Show that the relation $x, y \mapsto \|x = y\|$ is an equivalence relation, on any type.

16.5 Let $f : A \to X$ be a map. Construct an equivalence

$$\left(\sum_{(y:\mathsf{join\_power}_X(n,A))} f(x) = f^{*n}(y)\right) \simeq \left(\sum_{(y:A)} f(x) = f(y)\right)^{*n}$$

for any $x : A$.

16.6 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 $A \to \mathsf{is\_equiv}(f)$.

(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.

16.7 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 $f : A \to B$.

Lecture 17

Set quotients

17.1 The universal property of set quotients

Definition 17.1.1. Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation, for $A : \mathcal{U}$, and consider a map $q : A \to B$ where the type B is a set, for which we have

$$\textstyle\prod_{(x,y:A)} R(x,y) \to q(x) = q(y).$$

We will define a map

$$\mathsf{quotient\_restr} : (B \to X) \to \Big(\textstyle\sum_{(f:A\to X)}\prod_{(x,y:A)} R(x,y) \to (f(x) = f(y))\Big).$$

Construction. Let $h : B \to X$. Then we have $h \circ q : A \to X$, so it remains to show that

$$\textstyle\prod_{(x,y:A)} R(x,y) \to (h(q(x)) = h(q(y)))$$

Consider $x, y : A$ which are related by R. Then we have an identification $p : q(x) = q(y)$, so it follows that $\mathsf{ap}_h\,(p) : h(q(x)) = h(q(y))$. □

Definition 17.1.2. Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation, for $A : \mathcal{U}$, and consider a map $q : A \to B$ satisfying

$$\textstyle\prod_{(x,y:A)} R(x,y) \to q(x) = q(y),$$

where the type B is a set. We say that the map $q : A \to B$ satisfies the universal property of the **set quotient** A/R if for any set X the map

$$\mathsf{quotient\_restr} : (B \to X) \to \Big(\textstyle\sum_{(f:A\to X)}\prod_{(x,y:A)} R(x,y) \to (f(x) = f(y))\Big)$$

is an equivalence.

Lemma 17.1.3. *Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation, for $A : \mathcal{U}$, and consider a commuting triangle*

$$\begin{tikzcd}
A \arrow[rr, "q"] \arrow[dr, "R"'] & & U \arrow[dl, "m"] \\
& \mathsf{Prop}^A &
\end{tikzcd}$$

with $H : R \sim m \circ q$, where m is an embedding. Then we have

$$\textstyle\prod_{(x,y:A)} R(x,y) \to (q(x) = q(y)).$$

Theorem 17.1.4. *Let $R : A \to (A \to \mathsf{Prop})$ be an equivalence relation, for $A : \mathcal{U}$, and consider a commuting triangle*

$$\begin{tikzcd}
A \arrow[rr, "q"] \arrow[dr, "R"'] & & U \arrow[dl, "m"] \\
& \mathsf{Prop}^A &
\end{tikzcd}$$

with $H : R \sim m \circ q$, where m is an embedding. Then the following are equivalent:

(i) The embedding $m : U \to \mathsf{Prop}^A$ satisfies the universal property of the image of R.

(ii) The map $q : A \to U$ satisfies the universal property of the set quotient A/R.

Proof. Suppose $m : U \to \mathsf{Prop}^A$ satisfies the universal property of the image of R. Then it follows by Theorem 16.5.3 that the map $q : A \to U$ is surjective. Our goal is to prove that U satisfies the universal property of the set quotient A/R. □

Remark 17.1.5. Theorem 17.1.4 suggests that we can define the quotient of an equivalence relation R on a type A as the image of a map. However, the type Prop^A of which the quotient is a subtype is not a small type, even if A is a small type. Therefore it is not clear that the quotient A/R is essentially small, as it should be. Luckily, our construction of the image of a map allows us to show that the image is indeed essentially small, using the fact that Prop^A is locally small.

17.2 The construction of set quotients

Lemma 17.2.1. *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 17.2.2. *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 17.2.3. *Consider a type sequence*

$$\begin{tikzcd} A_0 \arrow[r, "f_0"] & A_1 \arrow[r, "f_1"] & A_2 \arrow[r, "f_2"] & \cdots \end{tikzcd}$$

where each A_n is essentially small. Then its sequential colimit is again essentially small.

Theorem 17.2.4. *For any map $f : A \to X$ from a small type A into a locally small type X, the image* $\mathrm{im}(f)$ *is an essentially small type.*

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.

Corollary 17.2.5. *For any small type family $B : A \to \mathcal{U}$, where A is small, the image* $\mathrm{im}(B)$ *is essentially small. We call* $\mathrm{im}(B)$ *the* ***univalent completion*** *of B.*

17.3 Connected components of types

17.4 Set truncation

Lemma 17.4.1. *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 17.4.2. 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 17.4.3. *For each type A, the set truncation satisfies the universal property of the set truncation.*

Exercises

17.1 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

$$\begin{tikzcd}
\sum_{(x,y:A)} R(x,y) \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A \arrow[d, "\mathsf{inr}"] \\
A \arrow[r, "\mathsf{inl}"'] & A \sqcup^R A.
\end{tikzcd}$$

17.2 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.

17.3 Show that the type of pointed 2-element sets

$$\textstyle\sum_{(X:\mathcal{U}_{\mathbf{2}})} X$$

is contractible.

17.4 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 3.2.5.

(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.

Lecture 18

Homotopy groups of types

18.1 Pointed types

Definition 18.1.1. (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:

$$\Pi^*_{(x:X)} P(x) :\equiv \textstyle\sum_{(f:\prod_{(x:X)} P(x))} f(*_X) = *_P$$

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 $p \cdot q^{-1}$.

Example 18.1.2. The circle $\mathbb{S}^1$ is a pointed type with base point $\mathsf{base} : \mathbb{S}^1$.

Example 18.1.3. If X is a pointed type, then in the suspension of X we have the canonical identification $\mathsf{merid}(*_X) : \mathsf{N} = \mathsf{S}$. Therefore we do not have to worry about whether to choose N or S as the base point of ΣX.

Remark 18.1.4. 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.

Definition 18.1.5. (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**

$$g \circ_* f :\equiv (g \circ f, \mathsf{ap}_g\,(p_f) \cdot p_g).$$

18.2 Loop spaces

Definition 18.2.1. 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 18.2.2. 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 $\mathsf{right\_inv}(p_f)$.

(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.

Theorem 18.2.3. *Consider two pointed types (X, x_0) and (Y, y_0). Then there is an equivalence*

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

Proof. Computing with the universal property of the suspension

$$\begin{aligned}\Sigma X \to_* Y &\simeq \textstyle\sum_{(y,y':Y)}(X \to (y = y')) \times (y' = y_0)\\ &\simeq \textstyle\sum_{(y:Y)} X \to (y = y_0)\\ &\simeq \textstyle\sum_{(f:X\to(y=y_0))} f(x_0) = \mathsf{refl}_{y_0}.\end{aligned}$$

In the last equivalence we used Exercise 6.5. □

18.3 Homotopy groups

In homotopy type theory we use 0-types to define groups.

Definition 18.3.1. A **group** $\mathcal{G}$ consists of a set G with a unit $e : G$, a multiplication $x, y \mapsto x \cdot y$, and an inverse operation $x \mapsto x^{-1}$ satisfying the **group laws**:

$$\begin{aligned}(x \cdot y) \cdot z &= x \cdot (y \cdot z) & x^{-1} \cdot x &= e\\ e \cdot x &= x & x \cdot x^{-1} &= e.\\ x \cdot e &= x\end{aligned}$$

Definition 18.3.2. 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 18.3.3. In Corollary 12.2.7 we established that $\Omega(\mathbb{S}^1) \simeq \mathbb{Z}$. It follows that

$$\pi_1(\mathbb{S}^1) = \mathbb{Z} \qquad \text{and} \qquad \pi_n(\mathbb{S}^1) = 0 \qquad \text{for } n \geq 2.$$

Furthermore, we have seen in **??** that $\|\mathbb{S}^1\|_0$ is contractible. Therefore we also have $\pi_0(\mathbb{S}^1) = 0$.

18.4 The Eckmann-Hilton argument

Given a diagram of identifications

$$\begin{tikzcd}
x \arrow[r, equal, bend left=60, "p"] \arrow[r, equal, "p'"] \arrow[r, equal, bend right=60, "p''"'] \arrow[r, phantom, shift left=3ex, "r \Downarrow"] \arrow[r, phantom, shift right=3ex, "r' \Downarrow"] & y
\end{tikzcd}$$

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 18.4.1. 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

$$\begin{tikzcd}
x \arrow[r, equal, bend left, "p"] \arrow[r, equal, bend right, "p'"'] \arrow[r, phantom, "r \Downarrow"] & y \arrow[r, equal, bend left, "q"] \arrow[r, equal, bend right, "q'"'] \arrow[r, phantom, "s \Downarrow"] & z.
\end{tikzcd}$$

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 18.4.2. *Horizontal concatenation satisfies the left and right unit laws.*

In the following lemma we establish the **interchange law** for horizontal and vertical concatenation.

Lemma 18.4.3. *Consider a diagram of the form*

$$\begin{tikzcd}
x \arrow[r, equal, bend left=60, "p"] \arrow[r, equal] \arrow[r, equal, bend right=60, "p''"'] \arrow[r, phantom, shift left=3ex, "r \Downarrow"] \arrow[r, phantom, shift right=3ex, "r' \Downarrow"] & y \arrow[r, equal, bend left=60, "q"] \arrow[r, equal] \arrow[r, equal, bend right=60, "q''"'] \arrow[r, phantom, shift left=3ex, "s \Downarrow"] \arrow[r, phantom, shift right=3ex, "s' \Downarrow"] & z.
\end{tikzcd}$$

Then there is an identification

$$(r \centerdot r') \centerdot_h (s \centerdot s') = (r \centerdot_h s) \centerdot (r' \centerdot_h s').$$

Proof. We use path induction on both r and r', followed by path induction on p. Then it suffices to show that

$$(\mathsf{refl}_{\mathsf{refl}_y} \centerdot \mathsf{refl}_{\mathsf{refl}_y}) \centerdot_h (s \centerdot s') = (\mathsf{refl}_{\mathsf{refl}_y} \centerdot_h s) \centerdot (\mathsf{refl}_{\mathsf{refl}_y} \centerdot_h s').$$

Using the computation rules, we see that this reduces to

$$s \centerdot s' = s \centerdot s',$$

which we have by reflexivity. □

Theorem 18.4.4. *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 $|r|_0 \cdot |s|_0 \equiv |r \centerdot s|_0$ and $|s|_0 \cdot |r|_0 \equiv |s \centerdot r|_0$, to see that it suffices to show that $r \centerdot s = s \centerdot r$, for every $r, s : \mathsf{refl}_x = \mathsf{refl}_x$. Using the unit laws and the interchange law, this is a simple computation:

$$\begin{aligned} r \centerdot s &= (r \centerdot_h \mathsf{refl}_x) \centerdot (\mathsf{refl}_x \centerdot_h s) \\ &= (r \centerdot \mathsf{refl}_x) \centerdot_h (\mathsf{refl}_x \centerdot s) \\ &= (\mathsf{refl}_x \centerdot r) \centerdot_h (s \centerdot \mathsf{refl}_x) \\ &= (\mathsf{refl}_x \centerdot_h s) \centerdot (r \centerdot_h \mathsf{refl}_x) \\ &= s \centerdot r. \end{aligned}$$

□

18.5 Simply connectedness of the 2-sphere

Definition 18.5.1. A pointed type X is said to be n**-connected** if its homotopy groups $\pi_i(X)$ are trivial for $i \leq n$. A 0-connected type is also just called **connected**, and a 1-connected type is also called **simply connected**.

We write $*$ for the base point of the sphere $\mathbb{S}^n$.

Theorem 18.5.2. *For any $n : \mathbb{N}$ and any family P of n-types over the $(n+2)$-sphere $\mathbb{S}^{n+2}$, the function*

$$\Big(\prod_{(x:\mathbb{S}^{n+2})} P(x)\Big) \to P(*)$$

given by $f \mapsto f(*)$, is an equivalence.

Corollary 18.5.3. *The 2-sphere is simply connected.*

Proof. Our goal is to show that $\pi_1(\mathbb{S}^2)$ is contractible. In other words, we have to show that $\|\Omega(\mathbb{S}^2)\|_0$ is contractible. We do this by constructing a term of type

$$\prod_{(t:\mathbb{S}^2)} \mathsf{is\_contr}(\|* = t\|_0).$$

First we note that

$$\prod_{(t:\mathbb{S}^2)} \|* = t\|_0$$

is equivalent to the type $\|* = *\|_0$, of which we have the term $|\mathsf{refl}_*|_0$. Thus we obtain a dependent function $\alpha : \prod_{(t:\mathbb{S}^2)} \|* = t\|_0$ equipped with $\alpha(*) = |\mathsf{refl}_*|_0$. Now we proceed to show that

$$\prod_{(t:\mathbb{S}^2)} \prod_{(p:\|*=t\|_0)} \alpha(t) = p$$

by the dependent universal property of 0-truncation. Therefore it suffices to construct a term of type

$$\prod_{(t:\mathbb{S}^2)} \prod_{(p:*=t)} \alpha(t) = |p|_0.$$

This is immediate by path induction and the fact that $\alpha(*) = |\mathsf{refl}_*|_0$. □

Exercises

18.1 Show that the type of pointed families over a pointed type (X, x) is equivalent to the type

$$\sum_{(Y:\mathcal{U}_*)} Y \to_* X.$$

18.2 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.

18.3 Construct by path induction a family of maps

$$\textstyle\prod_{(A,B:\mathcal{U})}\prod_{(a:A)}\prod_{(b:B)}((A,a)=(B,b)) \to \sum_{(e:A\simeq B)} e(a)=b,$$

and show that this map is an equivalence. In other words, an *identification of pointed types* is a base point preserving equivalence.

18.4 Let (A,a) and (B,b) be two pointed types. Construct by path induction a family of maps

$$\textstyle\prod_{(f,g:A\to B)}\prod_{(p:f(a)=b)}\prod_{(q:g(a)=b)}((f,p)=(g,q)) \to \sum_{(H:f\sim g)} p = H(a) \cdot q,$$

and show that this map is an equivalence. In other words, an *identification of pointed maps* is a base point preserving homotopy.

18.5 Show that if $A \leftarrow S \rightarrow B$ is a span of pointed types, then for any pointed type X the square

$$\begin{tikzcd}
(A \sqcup^S B \to_* X) \arrow[r] \arrow[d] & (B \to_* X) \arrow[d] \\
(A \to_* X) \arrow[r] & (S \to_* X)
\end{tikzcd}$$

is a pullback square.

18.6 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.

18.7 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

$$\begin{tikzcd}
(\Sigma(X') \to_* Y') \arrow[r, "\tau_{X',Y'}"] \arrow[d, "h\mapsto g\circ h\circ\Sigma(f)"'] & (X' \to_* \Omega(Y')) \arrow[d, "h\mapsto\Omega(g)\circ h\circ f"] \\
(\Sigma(X) \to_* Y) \arrow[r, "\tau_{X,Y}"'] & (X \to_* \Omega(Y))
\end{tikzcd}$$

18.8 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

$$\begin{tikzcd} \Omega(C) \arrow[r] \arrow[d] & \Omega(B) \arrow[d] \\ \Omega(A) \arrow[r] & \Omega(X). \end{tikzcd}$$

18.9 (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-trunctated.

It is consistent to assume that there are types for which all homotopy groups are trivial, but which aren't contractible nontheless. Such types are called **∞-connected**.

Lecture 19

The long exact sequence of homotopy groups

19.1 The long exact sequence

Definition 19.1.1. 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 19.1.2. *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

$$\begin{tikzcd}
\Omega(F) \arrow[r] \arrow[d, swap, "\Omega(i)"] & \mathbf{1} \arrow[d, "\mathsf{const}_{\mathsf{refl}_{b_0}}"] & \\
\Omega(E) \arrow[r, "\Omega(p)"] \arrow[d] & \Omega(B) \arrow[r] \arrow[d, swap, "\partial"] & \mathbf{1} \arrow[d, "\mathsf{const}_{y_0}"] \\
\mathbf{1} \arrow[r, swap, "\mathsf{const}_{x_0}"] & F \arrow[r, "i"] \arrow[d] & E \arrow[d, "p"] \\
& \mathbf{1} \arrow[r, swap, "\mathsf{const}_{b_0}"] & B.
\end{tikzcd}$$

□

Definition 19.1.3. 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 19.1.4. 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 19.1.5. *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

$$\textstyle\prod_{(x:\|E\|_0)}\big\|\mathsf{fib}_{\|i\|_0}(x)\big\|_{-1} \simeq \|p\|_0(x) = |b_0|_0.$$

By the universal property of 0-truncation it suffices to show that

$$\textstyle\prod_{(x:E)}\big\|\mathsf{fib}_{\|i\|_0}(|x|_0)\big\|_{-1} \simeq \|p\|_0(|x|_0) = |b_0|_0.$$

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

$$\begin{aligned}\mathsf{fib}_{\|i\|_0}(|x|_0) &\simeq \textstyle\sum_{(y:\|F\|_0)}\|i\|_0(y) = |x|_0\\ &\simeq \Big\|\textstyle\sum_{(y:F)}\|i\|_0(|y|_0) = |x|_0\Big\|_0\\ &\simeq \Big\|\textstyle\sum_{(y:F)}|i(y)|_0 = |x|_0\Big\|_0\\ &\simeq \Big\|\textstyle\sum_{(y:F)}\|i(y) = x\|_{-1}\Big\|_0.\end{aligned}$$

Therefore it follows that

$$\begin{aligned}\big\|\mathsf{fib}_{\|i\|_0}(|x|_0)\big\|_{-1} &\simeq \Big\|\textstyle\sum_{(y:F)}\|i(y) = x\|_{-1}\Big\|_{-1}\\ &\simeq \Big\|\textstyle\sum_{(y:F)}i(y) = x\Big\|_{-1}\end{aligned}$$

Now it suffices to show that $\big(\sum_{(y:F)} i(y) = x\big) \simeq p(x) = b_0$. 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 19.1.6. *Any fiber sequence $F \hookrightarrow E \twoheadrightarrow B$ induces a long exact sequence on homotopy groups*

$$\begin{tikzcd}
 & & \cdots \arrow[dll, rounded corners, to path={-- ([xshift=2ex]\tikztostart.east) |- ([yshift=-1ex]\tikztostart.south) -| ([xshift=-2ex]\tikztotarget.west) -- (\tikztotarget)}] \\
\pi_n(F) \arrow[r, "\pi_n(i)"] & \pi_n(E) \arrow[r, "\pi_n(p)"] & \pi_n(B) \arrow[dll, dotted, rounded corners, to path={-- ([xshift=2ex]\tikztostart.east) |- ([yshift=-1ex]\tikztostart.south) -| ([xshift=-2ex]\tikztotarget.west) -- (\tikztotarget)}] \\
\pi_1(F) \arrow[r, "\pi_1(i)"] & \pi_1(E) \arrow[r, "\pi_1(p)"] & \pi_1(B) \arrow[dll, rounded corners, to path={-- ([xshift=2ex]\tikztostart.east) |- ([yshift=-1ex]\tikztostart.south) -| ([xshift=-2ex]\tikztotarget.west) -- (\tikztotarget)}] \\
\pi_0(F) \arrow[r, "\pi_0(i)"] & \pi_0(E) \arrow[r, "\pi_0(p)"] & \pi_0(B)
\end{tikzcd}$$

19.2 The Hopf fibration

Our goal in this section is to construct the Hopf fibration, i.e. a fiber sequence

$$\mathbb{S}^1 \hookrightarrow \mathbb{S}^3 \twoheadrightarrow \mathbb{S}^2.$$

This fiber sequence involves the complex multiplication of the unit sphere in the complex number, which is a circle. Viewing the circle as a subspace of the complex numbers, we write 1 for the base point of the circle.

Definition 19.2.1. We define the **complex multiplication** operation

$$\mu_{\mathbb{C}} : \mathbb{S}^1 \to (\mathbb{S}^1 \to \mathbb{S}^1).$$

Construction. By the universal property of the circle, it is equialent to define

$$\begin{aligned} \mu_{\mathbb{C}}(1) &: \mathbb{S}^1 \to \mathbb{S}^1 \\ \mathsf{ap}_{\mu_{\mathbb{C}}}(\mathsf{loop}) &: \mu_{\mathbb{C}}(1) = \mu_{\mathbb{C}}(1). \end{aligned}$$

The function $\mu_{\mathbb{C}}(1)$ is multiplication by 1, which is the identity function. The type of $\mathsf{ap}_{\mu_{\mathbb{C}}}(\mathsf{loop})$ is equivalent to the type of homotopies

$$\mathsf{id}_{\mathbb{S}^1} \sim \mathsf{id}_{\mathbb{S}^1}.$$

We construct this homotopy by induction on $\mathbb{S}^1$. Therefore it suffices to construct

$$\begin{aligned} p &: 1 = 1 \\ q &: \mathsf{tr}_L(\mathsf{loop}, p) = p \end{aligned}$$

□

Lemma 19.2.2. *The complex multiplication operation $\mu_{\mathbb{C}}$ on the circle satisfies the unit laws*

$$\begin{aligned}\mathsf{left\_unit}_{\mathbb{C}}(x) &: \mu_{\mathbb{C}}(1,x) = x\\ \mathsf{right\_unit}_{\mathbb{C}}(x) &: \mu_{\mathbb{C}}(x,1) = x\\ \mathsf{coh\_unit}_{\mathbb{C}} &: \mathsf{left\_unit}_{\mathbb{C}}(1) = \mathsf{right\_unit}_{\mathbb{C}}(1),\end{aligned}$$

and the functions $\mu_{\mathbb{C}}(x,\mathord{-})$ and $\mu_{\mathbb{C}}(\mathord{-},y)$ are equivalences for each $x : \mathbb{S}^1$ and $y : \mathbb{S}^1$, respectively.

Lemma 19.2.3. *Both commuting squares in the diagram*

$$\begin{tikzcd}
\mathbb{S}^1 \arrow[d] & \mathbb{S}^1 \times \mathbb{S}^1 \arrow[l, "\mathsf{pr}_1"'] \arrow[r, "\mathsf{pr}_2"] \arrow[d, "\mu_{\mathbb{C}}"'] & \mathbb{S}^1 \arrow[d] \\
\mathbf{1} & \mathbb{S}^1 \arrow[l] \arrow[r] & \mathbf{1}
\end{tikzcd}$$

are pullback squares.

Corollary 19.2.4. *There is a fiber sequence*

$$\mathbb{S}^1 \hookrightarrow \mathbb{S}^1 * \mathbb{S}^1 \twoheadrightarrow \mathbb{S}^2.$$

Lemma 19.2.5. *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 19.2.6. *There is an equivalence $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$.*

Theorem 19.2.7. *There is a fiber sequence $\mathbb{S}^1 \hookrightarrow \mathbb{S}^3 \twoheadrightarrow \mathbb{S}^2$.*

Lemma 19.2.8. *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 19.2.9. *We have $\pi_2(\mathbb{S}^2) = \mathbb{Z}$, and for $k > 2$ we have $\pi_k(\mathbb{S}^2) = \pi_k(\mathbb{S}^3)$.*

Exercises

19.1 Give the 0-sphere $\mathbb{S}^0$ the structure of an H-space.

19.2 For any pointed type A, give $\Omega(A)$ the structure of an H-space.

19.3 Show that the type of (small) fiber sequences is equivalent to the type of quadruples (B, P, b_0, x_0), consisting of

$$\begin{aligned} B &: \mathcal{U} \\ P &: B \to \mathcal{U} \\ b_0 &: B \\ x_0 &: P(b_0). \end{aligned}$$

Bibliography

[1] G. Brunerie. “On the homotopy groups of spheres in homotopy type theory”. In: *ArXiv e-prints* (June 2016). arXiv: `1606.05916 [math.AT]`.

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

Index