Homotopy Type Theory

Basics

2.1 Types are higher groupoids

L 2.1.1. *For every type A and every $x, y : A$ there is a function*

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

*denoted $p \mapsto p^{-1}$, such that $\mathsf{refl}_x{}^{-1} \equiv \mathsf{refl}_x$ for each $x : A$. We call p^{-1} the **inverse** of p.*

L 2.1.2. *For every type A and every $x, y, z : A$ there is a function*

$$(x = y) \to (y = z) \to (x = z)$$

*written $p \mapsto q \mapsto p \cdot q$, such that $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$ for any $x : A$. We call $p \cdot q$ the **concatenation** or **composite** of p and q.*

Equality	Homotopy	∞-Groupoid
reflexivity	constant path	identity morphism
symmetry	inversion of paths	inverse morphism
transitivity	concatenation of paths	composition of morphisms

L 2.1.3. *Suppose $A : \mathcal{U}$, that $x, y, z, w : A$ and that $p : x = y$ and $q : y = z$ and $r : z = w$. We have the following:*

(i). $p = p \cdot \mathsf{refl}_y$ *and* $p = \mathsf{refl}_x \cdot p$.
(ii). $p^{-1} \cdot p = \mathsf{refl}_y$ *and* $p \cdot p^{-1} = \mathsf{refl}_x$.
(iii). $(p^{-1})^{-1} = p$.
(iv). $p \cdot (q \cdot r) = (p \cdot q) \cdot r$.

T 2.1.4 (Eckmann–Hilton). *The composition operation on the second loop space*

$$\Omega^2(A) \times \Omega^2(A) \to \Omega^2(A)$$

is commutative: $\alpha \cdot \beta = \beta \cdot \alpha$, for any $\alpha, \beta : \Omega^2(A)$.

D 2.1.5. A **pointed type** (A, a) is a type $A : \mathcal{U}$ together with a point $a : A$, called its **basepoint**. We write $\mathcal{U}_\bullet :\equiv \sum_{(A:\mathcal{U})} A$ for the type of pointed types in the universe $\mathcal{U}$.

D 2.1.6. Given a pointed type (A, a), we define the **loop space** of (A, a) to be the following pointed type:

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

An element of it will be called a **loop** at a. For $n : \mathbb{N}$, the **n-fold iterated loop space** $\Omega^n(A, a)$ of a pointed type (A, a) is defined recursively by:

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

An element of it will be called an **n-loop** or an **n-dimensional loop** at a.

2.2 Functions are functors

L 2.2.1. *Suppose that $f : A \to B$ is a function. Then for any $x, y : A$ there is an operation*

$$\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y)).$$

Moreover, for each $x : A$ we have $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$.

The notation ap_f can be read either as the application of f to a path, or as the action on paths of f.
We note that ap behaves functorially, in all the ways that one might expect.

L 2.2.2. *For functions $f : A \to B$ and $g : B \to C$ and paths $p : x =_A y$ and $q : y =_A z$, we have:*

(i). $\mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q)$.
(ii). $\mathsf{ap}_f(p^{-1}) = \mathsf{ap}_f(p)^{-1}$.
(iii). $\mathsf{ap}_g(\mathsf{ap}_f(p)) = \mathsf{ap}_{g \circ f}(p)$.
(iv). $\mathsf{ap}_{\mathsf{id}_A}(p) = p$.

2.3 Type families are fibrations

L 2.3.1 (Transport). *Suppose that P is a type family over A and that $p : x =_A y$. Then there is a function $p_* : P(x) \to P(y)$.*

Sometimes, it is necessary to notate the type family P in which the transport operation happens.

$$\mathsf{transport}^P(p, -) : P(x) \to P(y).$$

L 2.3.2 (Path lifting property). *Let $P : A \to \mathcal{U}$ be a type family over A and assume we have $u : P(x)$ for some $x : A$. Then for any $p : x = y$, we have*

$$\mathsf{lift}(u, p) : (x, u) = (y, p_*(u))$$

in $\sum_{(x:A)} P(x)$, such that $\mathsf{pr}_1(\mathsf{lift}(u, p)) = p$.

NB 2.3.3. Although we may think of a type family $P : A \to \mathcal{U}$ as like a fibration, it is generally not a good idea to say things like "the fibration $P : A \to \mathcal{U}$", since this sounds like we are talking about a fibration with base $\mathcal{U}$ and total space A. To repeat, when a type family $P : A \to \mathcal{U}$ is regarded as a fibration, the base is A and the total space is $\sum_{(x:A)} P(x)$.
We may also occasionally use other topological terminology when speaking about type families. For instance, we may refer to a dependent function $f : \prod_{(x:A)} P(x)$ as a **section** of the fibration P, and we may say that something happens **fiberwise** if it happens for each $P(x)$. For instance, a section $f : \prod_{(x:A)} P(x)$ shows that P is "fiberwise inhabited".

L 2.3.4 (Dependent map). *Suppose $f : \prod_{(x:A)} P(x)$; then we have a map*

$$\mathsf{apd}_f : \prod_{p:x=y} (p_*(f(x)) =_{P(y)} f(y)).$$

L 2.3.5. *If $P : A \to \mathcal{U}$ is defined by $P(x) :\equiv B$ for a fixed $B : \mathcal{U}$, then for any $x, y : A$ and $p : x = y$ and $b : B$ we have a path*

$$\mathsf{transportconst}^B_p(b) : \mathsf{transport}^P(p, b) = b.$$

L 2.3.8. *For $f : A \to B$ and $p : x =_A y$, we have*

$$\mathsf{apd}_f(p) = \mathsf{transportconst}^B_p(f(x)) \cdot \mathsf{ap}_f(p).$$

L 2.3.9. *Given $P : A \to \mathcal{U}$ with $p : x =_A y$ and $q : y =_A z$ while $u : P(x)$, we have*

$$q_*(p_*(u)) = (p \cdot q)_*(u).$$

L 2.3.10. *For a function $f : A \to B$ and a type family $P : B \to \mathcal{U}$, and any $p : x =_A y$ and $u : P(f(x))$, we have*

$$\mathsf{transport}^{P \circ f}(p, u) = \mathsf{transport}^P(\mathsf{ap}_f(p), u).$$

L 2.3.11. *For $P, Q : A \to \mathcal{U}$ and a family of functions $f : \prod_{(x:A)} P(x) \to Q(x)$, and any $p : x =_A y$ and $u : P(x)$, we have*

$$\mathsf{transport}^Q(p, f_x(u)) = f_y(\mathsf{transport}^P(p, u)).$$

2.4 Homotopies and equivalences

D 2.4.1. Let $f, g : \prod_{(x:A)} P(x)$ be two sections of a type family $P : A \to \mathcal{U}$. A **homotopy** from f to g is a dependent function of type

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

Note that a homotopy is not the same as an identification $(f = g)$.
However, in **??** we will introduce an axiom making homotopies and identifications "equivalent".
The following proofs are left to the reader.

L 2.4.2. *Homotopy is an equivalence relation on each dependent function type $\prod_{(x:A)} P(x)$. That is, we have elements of the types*

$$\prod_{f:\prod_{(x:A)} P(x)} (f \sim f)$$
$$\prod_{f,g:\prod_{(x:A)} P(x)} (f \sim g) \to (g \sim f)$$
$$\prod_{f,g,h:\prod_{(x:A)} P(x)} (f \sim g) \to (g \sim h) \to (f \sim h).$$

L 2.4.4. *Suppose $H : f \sim g$ is a homotopy between functions $f, g : A \to B$ and let $p : x =_A y$. Then we have*

$$H(x) \cdot g(p) = f(p) \cdot H(y).$$

We may also draw this as a commutative diagram:

$$\begin{tikzcd}
f(x) \arrow[r, equal, "f(p)"] \arrow[d, equal, "H(x)"'] & f(y) \arrow[d, equal, "H(y)"] \\
g(x) \arrow[r, equal, "g(p)"'] & g(y)
\end{tikzcd}$$

C 2.4.5. *Let $H : f \sim \mathsf{id}_A$ be a homotopy, with $f : A \to A$. Then for any $x : A$ we have*

$$H(f(x)) = f(H(x)).$$

$$\sum_{g:B\to A} \left((f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A)\right) \tag{2.4.6}$$

D 2.4.7. For a function $f : A \to B$, a **quasi-inverse** of f is a triple (g, α, β) consisting of a function $g : B \to A$ and homotopies $\alpha : f \circ g \sim \mathsf{id}_B$ and $\beta : g \circ f \sim \mathsf{id}_A$.

Thus, (2) is *the type of quasi-inverses of f*; we may denote it by $\mathsf{qinv}(f)$.

E 2.4.8. For any $p : x =_A y$ and $z : A$, the functions

$$(p \cdot -) : (y =_A z) \to (x =_A z) \qquad \text{and}$$
$$(- \cdot p) : (z =_A x) \to (z =_A y)$$

have quasi-inverses given by $(p^{-1} \cdot -)$ and $(- \cdot p^{-1})$, respectively;

E 2.4.9. For any $p : x =_A y$ and $P : A \to \mathcal{U}$, the function

$$\mathsf{transport}^P(p, -) : P(x) \to P(y)$$

has a quasi-inverse given by $\mathsf{transport}^P(p^{-1}, -)$; this follows from **??**.

$$\mathsf{isequiv}(f) :\equiv \left(\sum_{g:B\to A} (f \circ g \sim \mathsf{id}_B)\right) \times \left(\sum_{h:B\to A} (h \circ f \sim \mathsf{id}_A)\right). \tag{2.4.10}$$

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{isequiv}(f). \tag{2.4.11}$$

L 2.4.12. *Type equivalence is an equivalence relation on $\mathcal{U}$. More specifically:*

(i). For any A, the identity function id_A is an equivalence; hence $A \simeq A$.

(ii). For any $f : A \simeq B$, we have an equivalence $f^{-1} : B \simeq A$.

(iii). For any $f : A \simeq B$ and $g : B \simeq C$, we have $g \circ f : A \simeq C$.

2.5 The higher groupoid structure of type formers

2.6 Cartesian product types

$$(x =_{A\times B} y) \to (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)). \tag{2.6.1}$$

T 2.6.2. *For any x and y, the function (2) is an equivalence.*

T 2.6.3. *In the above situation, we have*

$$\mathsf{transport}^{A\times B}(p, x) =_{A(w)\times B(w)} (\mathsf{transport}^A(p, \mathsf{pr}_1 x), \mathsf{transport}^B(p, \mathsf{pr}_2 x)).$$

T 2.6.4. *In the above situation, given $x, y : A \times B$ and $p : \mathsf{pr}_1 x = \mathsf{pr}_1 y$ and $q : \mathsf{pr}_2 x = \mathsf{pr}_2 y$, we have*

$$f(\mathsf{pair}^=(p, q)) =_{(f(x)=f(y))} \mathsf{pair}^=(g(p), h(q)).$$

2.7 Σ-types

T 2.7.2. *Suppose that $P : A \to \mathcal{U}$ is a type family over a type A and let $w, w' : \sum_{(x:A)} P(x)$. Then there is an equivalence*

$$(w = w') \simeq \sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} p_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w').$$

C 2.7.3. *For $z : \sum_{(x:A)} P(x)$, we have $z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$.*

Note that the lifted path $\mathsf{lift}(u, p)$ of $p : x = y$ at $u : P(x)$ defined in **??** may be identified with the special case of the introduction form

$$\mathsf{pair}^=(p, \mathsf{refl}_{p_*(u)}) : (x, u) = (y, p_*(u)).$$

T 2.7.4. *Suppose we have type families*

$$P : A \to \mathcal{U} \qquad \textit{and} \qquad Q : \left(\sum_{x:A} P(x)\right) \to \mathcal{U}.$$

Then we can construct the type family over A defined by

$$x \mapsto \sum_{u:P(x)} Q(x, u).$$

For any path $p : x = y$ and any $(u, z) : \sum_{(u:P(x))} Q(x, u)$ we have

$$p_*((u, z)) = (p_*(u), \mathsf{pair}^=(p, \mathsf{refl}_{p_*(u)})_*(z)).$$

2.8 The unit type

T 2.8.1. *For any $x, y : \mathbf{1}$, we have $(x = y) \simeq \mathbf{1}$.*

2.9 Π-types and the function extensionality axiom

Given a type A and a type family $B : A \to \mathcal{U}$, consider the dependent function type $\prod_{(x:A)} B(x)$. We expect the type $f = g$ of paths from f to g in $\prod_{(x:A)} B(x)$ to be equivalent to the type of pointwise paths:

$$(f = g) \simeq \left(\prod_{x:A} (f(x) =_{B(x)} g(x))\right). \tag{2.9.1}$$

$$\mathsf{happly} : (f = g) \to \prod_{x:A} (f(x) =_{B(x)} g(x)) \tag{2.9.2}$$

A 2.9.3 (Function extensionality)**.** *For any A, B, f, and g, the function (2) is an equivalence.*

In particular, **??** implies that (2) has a quasi-inverse

$$\mathsf{funext} : \left(\prod_{x:A} (f(x) = g(x))\right) \to (f = g).$$

This function is also referred to as "function extensionality".

$$\mathsf{refl}_f = \mathsf{funext}(x \mapsto \mathsf{refl}_{f(x)})$$
$$\alpha^{-1} = \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x)^{-1})$$
$$\alpha \cdot \beta = \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x) \cdot \mathsf{happly}(\beta, x)).$$

Given a type X, a path $p : x_1 =_X x_2$, type families $A, B : X \to \mathcal{U}$, and a function $f : A(x_1) \to B(x_1)$, we have

$$\mathsf{transport}^{A\to B}(p, f) = \left(x \mapsto \mathsf{transport}^B(p, f(\mathsf{transport}^A(p^{-1}, x)))\right) \tag{2.9.4}$$

where $A \to B$ denotes abusively the type family $X \to \mathcal{U}$ defined by

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

Transporting dependent functions is similar, but more complicated. Suppose given X and p as before, type families $A : X \to \mathcal{U}$ and $B : \prod_{(x:X)} (A(x) \to \mathcal{U})$, and also a dependent function $f : \prod_{(a:A(x_1))} B(x_1, a)$. Then for $a : A(x_2)$, we have

$$\mathsf{transport}^{\Pi_A(B)}(p, f)(a) =$$

$$\mathsf{transport}^{\widehat{B}}\left((\mathsf{pair}^=(p^{-1}, \mathsf{refl}_{p^{-1}{}_*(a)}))^{-1}, f(\mathsf{transport}^A(p^{-1}, a))\right)$$

where $\Pi_A(B)$ and $\widehat{B}$ denote respectively the type families

$$\begin{aligned} \Pi_A(B) &:\equiv (x \mapsto \textstyle\prod_{(a:A(x))} B(x, a)) &&: X \to \mathcal{U} \\ \widehat{B} &:\equiv (w \mapsto B(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) &&: (\textstyle\sum_{(x:X)} A(x)) \to \mathcal{U}. \end{aligned} \tag{2.9.5}$$

L 2.9.6. *Given type families $A, B : X \to \mathcal{U}$ and $p : x =_X y$, and also $f : A(x) \to B(x)$ and $g : A(y) \to B(y)$, we have an equivalence*

$$(p_*(f) = g) \;\simeq\; \prod_{a:A(x)} (p_*(f(a)) = g(p_*(a))).$$

Moreover, if $q : p_*(f) = g$ corresponds under this equivalence to $\widehat{q}$, then for $a : A(x)$, the path

$$\mathsf{happly}(q, p_*(a)) : (p_*(f))(p_*(a)) = g(p_*(a))$$

is equal to the concatenated path $i \cdot j \cdot k$, where

- $i : (p_*(f))(p_*(a)) = p_*(f(p^{-1}{}_*(p_*(a))))$ *comes from* (2),
- $j : p_*(f(p^{-1}{}_*(p_*(a)))) = p_*(f(a))$ *comes from* **????**, *and*
- $k : p_*(f(a)) = g(p_*(a))$ *is* $\widehat{q}(a)$.

L 2.9.7. *Given type families $A : X \to \mathcal{U}$ and $B : \prod_{(x:X)} A(x) \to \mathcal{U}$ and $p : x =_X y$, and also $f : \prod_{(a:A(x))} B(x, a)$ and $g : \prod_{(a:A(y))} B(y, a)$, we have an equivalence*

$$(p_*(f) = g) \;\simeq\; \Big(\prod_{a:A(x)} \mathsf{transport}^{\widehat{B}}(\mathsf{pair}^{=}(p, \mathsf{refl}_{p_*(a)}), f(a)) = g(p_*(a))\Big)$$

with $\widehat{B}$ as in (2).

2.10 Universes and the univalence axiom

L 2.10.1. *For types $A, B : \mathcal{U}$, there is a certain function,*

$$\mathsf{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B), \tag{2.10.2}$$

defined in the proof.

A 2.10.3 (Univalence). *For any $A, B : \mathcal{U}$, the function* (2) *is an equivalence.*

- An introduction rule for $(A =_{\mathcal{U}} B)$, denoted ua for "univalence axiom":
$$\mathsf{ua} : (A \simeq B) \to (A =_{\mathcal{U}} B).$$
- The elimination rule, which is $\mathsf{idtoeqv}$,
$$\mathsf{idtoeqv} \equiv \mathsf{transport}^{X \mapsto X} : (A =_{\mathcal{U}} B) \to (A \simeq B).$$
- The propositional computation rule,
$$\mathsf{transport}^{X \mapsto X}(\mathsf{ua}(f), x) = f(x).$$
- The propositional uniqueness principle: for any $p : A = B$,
$$p = \mathsf{ua}(\mathsf{transport}^{X \mapsto X}(p)).$$

We can also identify the reflexivity, concatenation, and inverses of equalities in the universe with the corresponding operations on equivalences:

$$\mathsf{refl}_A = \mathsf{ua}(\mathsf{id}_A)$$
$$\mathsf{ua}(f) \cdot \mathsf{ua}(g) = \mathsf{ua}(g \circ f)$$
$$\mathsf{ua}(f)^{-1} = \mathsf{ua}(f^{-1}).$$

L 2.10.4. *For any type family $B : A \to \mathcal{U}$ and $x, y : A$ with a path $p : x = y$ and $u : B(x)$, we have*

$$\begin{aligned}\mathsf{transport}^B(p, u) &= \mathsf{transport}^{X \mapsto X}(\mathsf{ap}_B(p), u)\\ &= \mathsf{idtoeqv}(\mathsf{ap}_B(p))(u).\end{aligned}$$

2.11 Identity type

T 2.11.1. *If $f : A \to B$ is an equivalence, then for all $a, a' : A$, so is*

$$\mathsf{ap}_f : (a =_A a') \to (f(a) =_B f(a')).$$

L 2.11.2. *For any A and $a : A$, with $p : x_1 = x_2$, we have*

$$\begin{aligned}
&\mathsf{transport}^{x \mapsto (a = x)}(p, q) = q \cdot p && \text{for } q : a = x_1,\\
&\mathsf{transport}^{x \mapsto (x = a)}(p, q) = p^{-1} \cdot q && \text{for } q : x_1 = a,\\
&\mathsf{transport}^{x \mapsto (x = x)}(p, q) = p^{-1} \cdot q \cdot p && \text{for } q : x_1 = x_1.
\end{aligned}$$

T 2.11.3. *For $f, g : A \to B$, with $p : a =_A a'$ and $q : f(a) =_B g(a)$, we have*

$$\mathsf{transport}^{x \mapsto f(x) =_B g(x)}(p, q) =_{f(a') = g(a')} (\mathsf{ap}_f p)^{-1} \cdot q \cdot \mathsf{ap}_g p.$$

T 2.11.4. *Let $B : A \to \mathcal{U}$ and $f, g : \prod_{(x:A)} B(x)$, with $p : a =_A a'$ and $q : f(a) =_{B(a)} g(a)$. Then we have*

$$\mathsf{transport}^{x \mapsto f(x) =_{B(x)} g(x)}(p, q) = (\mathsf{apd}_f(p))^{-1} \cdot \mathsf{ap}_{(\mathsf{transport}^B p)}(q) \cdot \mathsf{apd}_g(p).$$

T 2.11.5. *For $p : a =_A a'$ with $q : a = a$ and $r : a' = a'$, we have*

$$(\mathsf{transport}^{x \mapsto (x = x)}(p, q) = r) \;\simeq\; (q \cdot p = p \cdot r).$$

2.12 Coproducts

T 2.12.1. *For all $x : A + B$ we have $(\mathsf{inl}(a_0) = x) \simeq \mathsf{code}(x)$.*

2.13 Natural numbers

We use the encode-decode method to characterize the path space of the natural numbers, which are also a positive type.

$$\mathsf{code} : \mathbb{N} \to \mathbb{N} \to \mathcal{U},$$

defined by double recursion over $\mathbb{N}$ as follows:

$$\begin{aligned}\mathsf{code}(0, 0) &:\equiv \mathbf{1}\\ \mathsf{code}(\mathsf{succ}(m), 0) &:\equiv \mathbf{0}\\ \mathsf{code}(0, \mathsf{succ}(n)) &:\equiv \mathbf{0}\\ \mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(n)) &:\equiv \mathsf{code}(m, n).\end{aligned}$$

We also define by recursion a dependent function $r : \prod_{(n:\mathbb{N})} \mathsf{code}(n, n)$, with

$$\begin{aligned}r(0) &:\equiv \star\\ r(\mathsf{succ}(n)) &:\equiv r(n).\end{aligned}$$

T 2.13.1. *For all $m, n : \mathbb{N}$ we have $(m = n) \simeq \mathsf{code}(m, n)$.*

2.14 Example: equality of structures

D 2.14.1. Given a type A, the type $\mathsf{SemigroupStr}$(A) of **semigroup structures** with carrier A is defined by

$$\mathsf{SemigroupStr}(A) :\equiv \sum_{(m:A \to A \to A)} \prod_{(x,y,z:A)} m(x, m(y, z)) = m(m(x, y), z).$$

A **semigroup** is a type together with such a structure:

$$\mathsf{Semigroup} :\equiv \sum_{A:\mathcal{U}} \mathsf{SemigroupStr}(A)$$

2.14.1 Lifting equivalences

$$\mathsf{transport}^{\mathsf{SemigroupStr}}(\mathsf{ua}(e)) : \mathsf{SemigroupStr}(A) \to \mathsf{SemigroupStr}(B).$$

Moreover, this map is an equivalence, because $\mathsf{transport}^C(\alpha)$ is always an equivalence with inverse $\mathsf{transport}^C(\alpha^{-1})$, see **????**.

2.15 Universal properties

$$(X \to A \times B) \to (X \to A) \times (X \to B) \tag{2.15.1}$$

defined by $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$.

T 2.15.2. (2) *is an equivalence.*

$$\Big(\prod_{x:X} (A(x) \times B(x))\Big) \to \Big(\prod_{x:X} A(x)\Big) \times \Big(\prod_{x:X} B(x)\Big) \tag{2.15.3}$$

defined as before by $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$.

T 2.15.4. (2) *is an equivalence.*

$$\Big(\prod_{x:X} \sum_{(a:A(x))} P(x, a)\Big) \to \Big(\sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x))\Big). \tag{2.15.5}$$

T 2.15.6. (2) *is an equivalence.*

For pullbacks, the expected explicit construction works: given $f : A \to C$ and $g : B \to C$, we define

$$A \times_C B :\equiv \sum_{(a:A)} \sum_{(b:B)} (f(a) = g(b)). \tag{2.15.7}$$

Exer. 2.1. Show that the three obvious proofs of **??** are pairwise equal.

Exer. 2.2. Show that the three equalities of proofs constructed in the previous exercise form a commutative triangle. In other words, if the three definitions of concatenation are denoted by $(p \cdot_1 q)$, $(p \cdot_2 q)$, and $(p \cdot_3 q)$, then the concatenated equality

$$(p \cdot_1 q) = (p \cdot_2 q) = (p \cdot_3 q)$$

is equal to the equality $(p \cdot_1 q) = (p \cdot_3 q)$.

Exer. 2.3. Give a fourth, different, proof of **??**, and prove that it is equal to the others.

Exer. 2.4. Define, by induction on n, a general notion of **n-dimensional path** in a type A, simultaneously with the type of boundaries for such paths.

Exer. 2.5. Prove that the functions (2) and (2) are inverse equivalences.

Exer. 2.6. Prove that if $p : x = y$, then the function $(p \cdot -) : (y = z) \to (x = z)$ is an equivalence.

Exer. 2.7. State and prove a generalization of **??** from cartesian products to Σ-types.

Exer. 2.8. State and prove an analogue of **??** for coproducts.

Exer. 2.9. Prove that coproducts have the expected universal property,

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

Exer. 2.10. Prove that Σ-types are "associative", in that for any $A : \mathcal{U}$ and families $B : A \to \mathcal{U}$ and $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}$, we have

$$\Big(\sum_{(x:A)} \sum_{(y:B(x))} C((x,y))\Big) \simeq \Big(\sum_{p:\sum_{(x:A)} B(x)} C(p)\Big).$$

Exer. 2.11. A (homotopy) **commutative square**

$$\begin{tikzcd} P \arrow[r, "h"] \arrow[d, "k"'] & A \arrow[d, "f"] \\ B \arrow[r, "g"'] & C \end{tikzcd}$$

consists of functions f, g, h, and k as shown, together with a path $f \circ h = g \circ k$. Note that this is exactly an element of the pullback $(P \to A) \times_{P \to C} (P \to B)$ as defined in (2). A commutative square is called a (homotopy) **pullback square** if for any X, the induced map

$$(X \to P) \to (X \to A) \times_{(X \to C)} (X \to B)$$

is an equivalence.

Exer. 2.12. Suppose given two commutative squares

$$\begin{tikzcd} A \arrow[r] \arrow[d] & C \arrow[r] \arrow[d] & E \arrow[d] \\ B \arrow[r] & D \arrow[r] & F \end{tikzcd}$$

and suppose that the right-hand square is a pullback square. Prove that the left-hand square is a pullback square if and only if the outer rectangle is a pullback square.

Exer. 2.13. Show that $(\mathbf{2} \simeq \mathbf{2}) \simeq \mathbf{2}$.

Exer. 2.14. Suppose we add to type theory the *equality reflection rule* which says that if there is an element $p : x = y$, then in fact $x \equiv y$. Prove that for any $p : x = x$ we have $p \equiv \mathsf{refl}_x$. (This implies that every type is a *set* in the sense to be introduced in **??**; see **??**.)

Exer. 2.15. Show that **??** can be strengthened to

$$\mathsf{transport}^B(p, -) =_{B(x) \to B(y)} \mathsf{idtoeqv}(\mathsf{ap}_B(p))$$

without using function extensionality. (In this and other similar cases, the apparently weaker formulation has been chosen for readability and consistency.)

Exer. 2.16. Suppose that rather than function extensionality (**??**), we suppose only the existence of an element

$$\mathsf{funext} : \prod_{(A:\mathcal{U})} \prod_{(B:A \to \mathcal{U})} \prod_{(f,g:\prod_{(x:A)} B(x))} (f \sim g) \to (f = g)$$

(with no relationship to happly assumed). Prove that in fact, this is sufficient to imply the whole function extensionality axiom (that happly is an equivalence). This is due to Voevodsky; its proof is tricky and may require concepts from later chapters.

Exer. 2.17.

(i). Show that if $A \simeq A'$ and $B \simeq B'$, then $(A \times B) \simeq (A' \times B')$.

(ii). Give two proofs of this fact, one using univalence and one not using it, and show that the two proofs are equal.

(iii). Formulate and prove analogous results for the other type formers: Σ, $\to$, Π, and $+$.

Exer. 2.18. State and prove a version of **??** for dependent functions.

Homotopy Type Theory

Sets and logic

3.1 Sets and n-types

D 3.1.1. A type A is a **set** if for all $x, y : A$ and all $p, q : x = y$, we have $p = q$.

More precisely, the proposition $\mathsf{isSet}(A)$ is defined to be the type

$$\mathsf{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q).$$

E 3.1.2. The type $\mathbf{1}$ is a set. For any $x, y : \mathbf{1}$ the type $(x = y)$ is equivalent to $\mathbf{1}$. Since any two elements of $\mathbf{1}$ are equal, this implies that any two elements of $x = y$ are equal.

E 3.1.3. The type $\mathbf{0}$ is a set, for given any $x, y : \mathbf{0}$ we may deduce anything we like, by the induction principle of $\mathbf{0}$.

E 3.1.4. The type $\mathbb{N}$ of natural numbers is also a set. Since all equality types $x =_{\mathbb{N}} y$ are equivalent to either $\mathbf{1}$ or $\mathbf{0}$, and any two inhabitants of $\mathbf{1}$ or $\mathbf{0}$ are equal.

Most of the type forming operations we have considered so far also preserve sets.

E 3.1.5. If A and B are sets, then so is $A \times B$. For given $x, y : A \times B$ and $p, q : x = y$, then we have $p = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(p), \mathsf{ap}_{\mathsf{pr}_2}(p))$ and $q = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(q), \mathsf{ap}_{\mathsf{pr}_2}(q))$. But $\mathsf{ap}_{\mathsf{pr}_1}(p) = \mathsf{ap}_{\mathsf{pr}_1}(q)$ since A is a set, and $\mathsf{ap}_{\mathsf{pr}_2}(p) = \mathsf{ap}_{\mathsf{pr}_2}(q)$ since B is a set; hence $p = q$.
Similarly, if A is a set and $B : A \to \mathcal{U}$ is such that each $B(x)$ is a set, then $\sum_{(x:A)} B(x)$ is a set.

E 3.1.6. If A is *any* type and $B : A \to \mathcal{U}$ is such that each $B(x)$ is a set, then the type $\prod_{(x:A)} B(x)$ is a set. For suppose $f, g : \prod_{(x:A)} B(x)$ and $p, q : f = g$. By function extensionality, we have

$$p = \mathsf{funext}(x \mapsto \mathsf{happly}(p, x)) \quad \text{and} \quad q = \mathsf{funext}(x \mapsto \mathsf{happly}(q, x)).$$

But for any $x : A$, we have

$$\mathsf{happly}(p, x) : f(x) = g(x) \qquad \text{and} \qquad \mathsf{happly}(q, x) : f(x) = g(x),$$

so since $B(x)$ is a set we have $\mathsf{happly}(p, x) = \mathsf{happly}(q, x)$. Now using function extensionality again, the dependent functions $(x \mapsto \mathsf{happly}(p, x))$ and $(x \mapsto \mathsf{happly}(q, x))$ are equal, and hence (applying $\mathsf{ap}_{\mathsf{funext}}$) so are p and q.

D 3.1.7. A type A is a **1-type** if for all $x, y : A$ and $p, q : x = y$ and $r, s : p = q$, we have $r = s$.

L 3.1.8. *If A is a set (that is, $\mathsf{isSet}(A)$ is inhabited), then A is a 1-type.*

3.2 Propositions as types?

NB 3.2.1. (Statement) If for all $x : X$ there exists an $a : A(x)$ such that $P(x, a)$, then there exists a function $g : \prod_{(x:A)} A(x)$ such that for all $x : X$ we have $P(x, g(x))$".
This looks like the classical *axiom of choice*, is always true under this reading.

NB 3.2.2. The classical *law of double negation* and *law of excluded middle* are incompatible with the univalence axiom.

T 3.2.3. *It is not the case that for all $A : \mathcal{U}$ we have $\neg(\neg A) \to A$.*

NB 3.2.4. For any A, $\neg\neg\neg A \to \neg A$ for any A.

C 3.2.5. *It is not the case that for all $A : \mathcal{U}$ we have $A + (\neg A)$.*

3.3 Mere propositions

D 3.3.1. A type P is a **mere proposition** if for all $x, y : P$ we have $x = y$.

Specifically, for any $P : \mathcal{U}$, the type $\mathsf{isProp}(P)$ is defined to be

$$\mathsf{isProp}(P) :\equiv \prod_{x,y:P} (x = y).$$

L 3.3.2. *If P is a mere proposition and $x_0 : P$, then $P \simeq \mathbf{1}$.*

L 3.3.3. *If P and Q are mere propositions such that $P \to Q$ and $Q \to P$, then $P \simeq Q$.*

L 3.3.4. *Every mere proposition is a set.*

L 3.3.5. *For any type A, the types $\mathsf{isProp}(A)$ and $\mathsf{isSet}(A)$ are mere propositions.*

3.4 Classical vs. intuitionistic logic

With the notion of mere proposition in hand, we can now give the proper formulation of the **law of excluded middle** in homotopy type theory:

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (A + \neg A)\Big). \tag{3.4.1}$$

Similarly, the **law of double negation** is

$$\prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (\neg\neg A \to A)\Big). \tag{3.4.2}$$

D 3.4.3.

(i). A type A is called **decidable** if $A + \neg A$.

(ii). Similarly, a type family $B : A \to \mathcal{U}$ is **decidable** if

$$\prod_{a:A} (B(a) + \neg B(a)).$$

(iii). In particular, A has **decidable equality** if

$$\prod_{a,b:A} ((a = b) + \neg(a = b)).$$

3.5 Subsets and propositional resizing

L 3.5.1. *Suppose $P : A \to \mathcal{U}$ is a type family such that $P(x)$ is a mere proposition for all $x : A$. If $u, v : \sum_{(x:A)} P(x)$ are such that $\mathsf{pr}_1(u) = \mathsf{pr}_1(v)$, then $u = v$.*

For instance, recall that

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{isequiv}(f),$$

where each type $\mathsf{isequiv}(f)$ was supposed to be a mere proposition. It follows that if two equivalences have equal underlying functions, then they are equal as equivalences.
If $P : A \to \mathcal{U}$ is a family of mere propositions (i.e. each $P(x)$ is a mere proposition), we may write

$$\{\, x : A \mid P(x) \,\} \tag{3.5.2}$$

as an alternative notation for $\sum_{(x:A)} P(x)$. We may define the "subuniverses" of sets and of mere propositions in a universe $\mathcal{U}$:

$$\mathsf{Set}_{\mathcal{U}} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isSet}(A) \,\},$$
$$\mathsf{Prop}_{\mathcal{U}} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isProp}(A) \,\}.$$

An element of $\mathsf{Set}_{\mathcal{U}}$ is a type $A : \mathcal{U}$ together with evidence $s : \mathsf{isSet}(A)$, and similarly for $\mathsf{Prop}_{\mathcal{U}}$.

A 3.5.3 (Propositional resizing). *The map $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ is an equivalence.*

With propositional resizing, we can define the power set to be

$$\mathcal{P}(A) :\equiv (A \to \Omega),$$

which is then independent of $\mathcal{U}$.

3.6 The logic of mere propositions

E 3.6.1. If A and B are mere propositions, so is $A \times B$. This is easy to show using the characterization of paths in products, just like **??** but simpler. Thus, the connective "and" preserves mere propositions.

E 3.6.2. If A is any type and $B : A \to \mathcal{U}$ is such that for all $x : A$, the type $B(x)$ is a mere proposition, then $\prod_{(x:A)} B(x)$ is a mere proposition.
The proof is just like **??** but simpler: given $f, g : \prod_{(x:A)} B(x)$, for any $x : A$ we have $f(x) = g(x)$ since $B(x)$ is a mere proposition. But then by function extensionality, we have $f = g$.
In particular, if B is a mere proposition, then so is $A \to B$ regardless of what A is. In even more particular, since $\mathbf{0}$ is a mere proposition, so is $\neg A \equiv (A \to \mathbf{0})$. Thus, the connectives "implies" and "not" preserve mere propositions, as does the quantifier "for all".

3.7 Propositional truncation

The *propositional truncation*, also called the *(-1)-truncation*, *bracket type*, or *squash type*, is an additional type former which "squashes" or "truncates" a type down to a mere proposition, forgetting all information contained in inhabitants of that type other than their existence.
More precisely, for any type A, there is a type $\|A\|$. It has two constructors:

- For any $a : A$ we have $|a| : \|A\|$.
- For any $x, y : \|A\|$, we have $x = y$.

The recursion principle of $\|A\|$ says that:

- If B is a mere proposition and we have $f : A \to B$, then there is an induced $g : \|A\| \to B$ such that $g(|a|) \equiv f(a)$ for all $a : A$.

D 3.7.1. We define **traditional logical notation** using truncation as follows, where P and Q denote mere propositions (or families thereof):

$$\begin{aligned}
\top &:\equiv \mathbf{1}\\
\bot &:\equiv \mathbf{0}\\
P \wedge Q &:\equiv P \times Q\\
P \Rightarrow Q &:\equiv P \to Q\\
P \Leftrightarrow Q &:\equiv P = Q\\
\neg P &:\equiv P \to \mathbf{0}\\
P \vee Q &:\equiv \|P + Q\|\\
\forall (x : A).\, P(x) &:\equiv \prod_{x:A} P(x)\\
\exists (x : A).\, P(x) &:\equiv \left\|\sum_{x:A} P(x)\right\|
\end{aligned}$$

The notations $\wedge$ and $\vee$ are also used in homotopy theory for the smash product and the wedge of pointed spaces.

$$\begin{aligned}
\{\, x : A \mid P(x) \,\} \cap \{\, x : A \mid Q(x) \,\} &:\equiv \{\, x : A \mid P(x) \wedge Q(x) \,\},\\
\{\, x : A \mid P(x) \,\} \cup \{\, x : A \mid Q(x) \,\} &:\equiv \{\, x : A \mid P(x) \vee Q(x) \,\},\\
A \setminus \{\, x : A \mid P(x) \,\} &:\equiv \{\, x : A \mid \neg P(x) \,\}.
\end{aligned}$$

Of course, in the absence of LEM, the latter are not "complements" in the usual sense: we may not have $B \cup (A \setminus B) = A$ for every subset B of A.

3.8 The axiom of choice

$$A : X \to \mathcal{U} \qquad \text{and} \qquad P : \prod_{x:X} A(x) \to \mathcal{U},$$

and moreover that

- X is a set,
- $A(x)$ is a set for all $x : X$, and
- $P(x, a)$ is a mere proposition for all $x : X$ and $a : A(x)$.

The **axiom of choice** AC asserts that under these assumptions,

$$\left(\prod_{x:X} \left\| \sum_{a:A(x)} P(x,a) \right\|\right) \to \left\| \sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x)) \right\|. \tag{3.8.1}$$

Of course, this is a direct translation of (3) where we read "there exists $x : A$ such that $B(x)$" as $\left\|\sum_{(x:A)} B(x)\right\|$, so we could have written the statement in the familiar logical notation as

$$\Big(\forall (x : X).\, \exists (a : A(x)).\, P(x,a)\Big) \Rightarrow \Big(\exists (g : \prod_{x:X} A(x)).\, \forall (x : X).\, P(x, g(x))\Big).$$

L 3.8.2. *The axiom of choice (3) is equivalent to the statement that for any set X and any $Y : X \to \mathcal{U}$ such that each $Y(x)$ is a set, we have*

$$\left(\prod_{x:X} \|Y(x)\|\right) \to \left\|\prod_{x:X} Y(x)\right\|. \tag{3.8.3}$$

NB 3.8.4. The right side of (3) always implies the left. Since both are mere propositions, by **??** the axiom of choice is also equivalent to asking for an equivalence

$$\left(\prod_{x:X} \|Y(x)\|\right) \simeq \left\|\prod_{x:X} Y(x)\right\|$$

L 3.8.5. *There exists a type X and a family $Y : X \to \mathcal{U}$ such that each $Y(x)$ is a set, but such that (3) is false.*

3.9 The principle of unique choice

L 3.9.1. *If P is a mere proposition, then $P \simeq \|P\|$.*

C 3.9.2 (The principle of unique choice)**.** *Suppose a type family $P : A \to \mathcal{U}$ such that*

(i). For each x, the type $P(x)$ is a mere proposition, and
(ii). For each x we have $\|P(x)\|$.

Then we have $\prod_{(x:A)} P(x)$.

3.11 Contractibility

D 3.11.1. A type A is **contractible**, or a **singleton**, if there is $a : A$, called the **center of contraction**, such that $a = x$ for all $x : A$. We denote the specified path $a = x$ by contr_x.

In other words, the type $\mathsf{isContr}(A)$ is defined to be

$$\mathsf{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x).$$

L 3.11.2. *For a type A, the following are logically equivalent.*

(i). A is contractible in the sense of **??**.
(ii). A is a mere proposition, and there is a point $a : A$.
(iii). A is equivalent to $\mathbf{1}$.

L 3.11.3. *For any type A, the type $\mathsf{isContr}(A)$ is a mere proposition.*

C 3.11.4. *If A is contractible, then so is $\mathsf{isContr}(A)$.*

L 3.11.5. *If $P : A \to \mathcal{U}$ is a type family such that each $P(a)$ is contractible, then $\prod_{(x:A)} P(x)$ is contractible.*

Of course, if A is equivalent to B and A is contractible, then so is B. More generally, it suffices for B to be a *retract* of A. By definition, a **retraction** is a function $r : A \to B$ such that there exists a function $s : B \to A$, called its **section**, and a homotopy $\epsilon : \prod_{(y:B)} (r(s(y)) = y)$; then we say that B is a **retract**of A.

L 3.11.6. *If B is a retract of A, and A is contractible, then so is B.*

L 3.11.7. *For any A and any $a : A$, the type $\sum_{(x:A)} (a = x)$ is contractible.*

L 3.11.8. *Let $P : A \to \mathcal{U}$ be a type family.*

(i). If each $P(x)$ is contractible, then $\sum_{(x:A)} P(x)$ is equivalent to A.
(ii). If A is contractible with center a, then $\sum_{(x:A)} P(x)$ is equivalent to $P(a)$.

L 3.11.9. *A type A is a mere proposition if and only if for all $x, y : A$, the type $x =_A y$ is contractible.*

Exer. 3.19. Prove that if $A \simeq B$ and A is a set, then so is B.

Exer. 3.20. Prove that if A and B are sets, then so is $A + B$.

Exer. 3.21. Prove that if A is a set and $B : A \to \mathcal{U}$ is a type family such that $B(x)$ is a set for all $x : A$, then $\sum_{(x:A)} B(x)$ is a set.

Exer. 3.22. Show that A is a mere proposition if and only if $A \to A$ is contractible.

Exer. 3.23. Show that $\mathsf{isProp}(A) \simeq (A \to \mathsf{isContr}(A))$.

Exer. 3.24. Show that if A is a mere proposition, then so is $A + (\neg A)$. Thus, there is no need to insert a propositional truncation in (3).

Exer. 3.25. More generally, show that if A and B are mere propositions and $\neg(A \times B)$, then $A + B$ is also a mere proposition.

Exer. 3.26. Show that if A and B are mere propositions such that $A \to B$ and $B \to A$, then $A \simeq B$.

Exer. 3.27. Show that for any type A, the types $\mathsf{isProp}(A)$ and $\mathsf{isSet}(A)$ are mere propositions.

Exer. 3.28. Show that if A is already a mere proposition, then $A \simeq \|A\|$.

Exer. 3.29. Assuming that some type $\mathsf{isequiv}(f)$ satisfies conditions (–(of **??**, show that the type $\|\mathsf{qinv}(f)\|$ satisfies the same conditions and is equivalent to $\mathsf{isequiv}(f)$.

Exer. 3.30. Show that if LEM holds, then the type $\mathsf{Prop} :\equiv \sum_{(A:\mathcal{U})} \mathsf{isProp}(A)$ is equivalent to $\mathbf{2}$.

Exer. 3.31. Show that if $\mathcal{U}_{i+1}$ satisfies LEM, then the canonical inclusion $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ is an equivalence.

Exer. 3.32. Show that it is not the case that for all $A : \mathcal{U}$ we have $\|A\| \to A$. (However, there can be particular types for which $\|A\| \to A$. **??** implies that $\mathsf{qinv}(f)$ is such.)

Exer. 3.33. Show that if LEM holds, then for all $A : \mathcal{U}$ we have $\|(\|A\| \to A)\|$. (This property is a very simple form of the axiom of choice, which can fail in the absence of LEM; see [?].)

Exer. 3.34. We showed in **??** that the following naive form of LEM is inconsistent with univalence:

$$\prod_{A:\mathcal{U}} (A + (\neg A))$$

In the absence of univalence, this axiom is consistent. However, show that it implies the axiom of choice (3).

Exer. 3.35. Show that assuming LEM, the double negation $\neg\neg A$ has the same universal property as the propositional truncation $\|A\|$, and is therefore equivalent to it. Thus, under LEM, the propositional truncation can be defined rather than taken as a separate type former.

Exer. 3.36. Show that if we assume propositional resizing as in **??**, then the type

$$\prod_{P:\mathsf{Prop}} \big((A \to P) \to P\big)$$

has the same universal property as $\|A\|$. Thus, we can also define the propositional truncation in this case.

Exer. 3.37. Assuming LEM, show that double negation commutes with universal quantification of mere propositions over sets. That is, show that if X is a set and each $Y(x)$ is a mere proposition, then LEM implies

$$\Big(\prod_{x:X} \neg\neg Y(x)\Big) \simeq \Big(\neg\neg \prod_{x:X} Y(x)\Big). \qquad (3.11.10)$$

Observe that if we assume instead that each $Y(x)$ is a set, then (3) becomes equivalent to the axiom of choice (3).

Exer. 3.38. Show that the rules for the propositional truncation given in **??** are sufficient to imply the following induction principle: for any type family $B : \|A\| \to \mathcal{U}$ such that each $B(x)$ is a mere proposition, if for every $a : A$ we have $B(|a|)$, then for every $x : \|A\|$ we have $B(x)$.

Exer. 3.39. Show that the law of excluded middle (3) and the law of double negation (3) are logically equivalent.

Exer. 3.40. Suppose $P : \mathbb{N} \to \mathcal{U}$ is a decidable family (see **????**) of mere propositions. Prove that

$$\Big\|\sum_{n:\mathbb{N}} P(n)\Big\| \to \sum_{n:\mathbb{N}} P(n).$$

Exer. 3.41. Prove **??**(: if A is contractible with center a, then $\sum_{(x:A)} P(x)$ is equivalent to $P(a)$.

Exer. 3.42. Prove that $\mathsf{isProp}(P) \simeq (P \simeq \|P\|)$.

Exer. 3.43. As in classical set theory, the finite version of the axiom of choice is a theorem. Prove that the axiom of choice (3) holds when X is a finite type $\mathsf{Fin}(n)$ (as defined in **??**).

Exer. 3.44. Show that the conclusion of **??** is true if $P : \mathbb{N} \to \mathcal{U}$ is any decidable family.

Homotopy Type Theory

Equivalences

Recall that we wanted $\mathsf{isequiv}(f)$ to have the following properties, which we restate here:

(i). $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$.
(ii). $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$.
(iii). $\mathsf{isequiv}(f)$ is a mere proposition.

Here $\mathsf{qinv}(f)$ denotes the type of quasi-inverses to f:

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

By function extensionality, it follows that $\mathsf{qinv}(f)$ is equivalent to the type

$$\sum_{g:B\to A} ((f\circ g = \mathsf{id}_B) \times (g \circ f = \mathsf{id}_A)).$$

We will define three different types having properties (–(, which we call

- half adjoint equivalences,
- bi-invertible maps, and
- contractible functions.

4.1 Quasi-inverses

L 4.1.1. *If $f : A \to B$ is such that $\mathsf{qinv}(f)$ is inhabited, then*

$$\mathsf{qinv}(f) \simeq \Big(\prod_{x:A}(x = x)\Big).$$

L 4.1.2. *Suppose we have a type A with $a : A$ and $q : a = a$ such that*

(i). The type $a = a$ is a set.
(ii). For all $x : A$ we have $\|a = x\|$.
(iii). For all $p : a = a$ we have $p \cdot q = q \cdot p$.

Then there exists $f : \prod_{(x:A)}(x = x)$ with $f(a) = q$.

T 4.1.3. *There exist types A and B and a function $f : A \to B$ such that $\mathsf{qinv}(f)$ is not a mere proposition.*

4.2 Half adjoint equivalences

D 4.2.1. A function $f : A \to B$ is a **half adjoint equivalence** if there are $g : B \to A$ and homotopies $\eta : g \circ f \sim \mathsf{id}_A$ and $\epsilon : f \circ g \sim \mathsf{id}_B$ such that there exists a homotopy

$$\tau : \prod_{x:A} f(\eta x) = \epsilon(fx).$$

L 4.2.2. *For functions $f : A \to B$ and $g : B \to A$ and homotopies $\eta : g \circ f \sim \mathsf{id}_A$ and $\epsilon : f \circ g \sim \mathsf{id}_B$, the following conditions are logically equivalent:*

- $\prod_{(x:A)} f(\eta x) = \epsilon(fx)$
- $\prod_{(y:B)} g(\epsilon y) = \eta(gy)$

T 4.2.3. *For any $f : A \to B$ we have $\mathsf{qinv}(f) \to \mathsf{ishae}(f)$.*

D 4.2.4. The **fiber** of a map $f : A \to B$ over a point $y : B$ is

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

In homotopy theory, this is what would be called the *homotopy fiber* of f. The path lemmas in **??** yield the following characterization of paths in fibers:

L 4.2.5. *For any $f : A \to B$, $y : B$, and $(x,p), (x',p') : \mathsf{fib}_f(y)$, we have*

$$((x,p) = (x',p')) \simeq \Big(\sum_{\gamma:x=x'} f(\gamma)\cdot p' = p\Big)$$

T 4.2.6. *If $f : A \to B$ is a half adjoint equivalence, then for any $y : B$ the fiber $\mathsf{fib}_f(y)$ is contractible.*

D 4.2.7. Given a function $f : A \to B$, we define the types

$$\mathsf{linv}(f) :\equiv \sum_{g:B\to A}(g\circ f \sim \mathsf{id}_A)$$
$$\mathsf{rinv}(f) :\equiv \sum_{g:B\to A}(f\circ g \sim \mathsf{id}_B)$$

of **left inverses** and **right inverses** to f, respectively. We call f **left invertible** if $\mathsf{linv}(f)$ is inhabited, and similarly **right invertible** if $\mathsf{rinv}(f)$ is inhabited.

L 4.2.8. *If $f : A \to B$ has a quasi-inverse, then so do*

$$(f\circ -) : (C \to A) \to (C \to B)$$
$$(-\circ f) : (B \to C) \to (A \to C).$$

L 4.2.9. *If $f : A \to B$ has a quasi-inverse, then the types $\mathsf{rinv}(f)$ and $\mathsf{linv}(f)$ are contractible.*

D 4.2.10. For $f : A \to B$, a left inverse $(g,\eta) : \mathsf{linv}(f)$, and a right inverse $(g,\epsilon) : \mathsf{rinv}(f)$, we denote

$$\mathsf{lcoh}_f(g,\eta) :\equiv \sum_{(\epsilon:f\circ g\sim \mathsf{id}_B)} \prod_{(y:B)} g(\epsilon y) = \eta(gy),$$
$$\mathsf{rcoh}_f(g,\epsilon) :\equiv \sum_{(\eta:g\circ f\sim \mathsf{id}_A)} \prod_{(x:A)} f(\eta x) = \epsilon(fx).$$

L 4.2.11. *For any f, g, ϵ, η, we have*

$$\mathsf{lcoh}_f(g,\eta) \simeq \prod_{y:B}(fgy, \eta(gy)) =_{\mathsf{fib}_g(gy)} (y, \mathsf{refl}_{gy}),$$
$$\mathsf{rcoh}_f(g,\epsilon) \simeq \prod_{x:A}(gfx, \epsilon(fx)) =_{\mathsf{fib}_f(fx)} (x, \mathsf{refl}_{fx}).$$

L 4.2.12. *If f is a half adjoint equivalence, then for any $(g,\epsilon) : \mathsf{rinv}(f)$, the type $\mathsf{rcoh}_f(g,\epsilon)$ is contractible.*

T 4.2.13. *For any $f : A \to B$, the type $\mathsf{ishae}(f)$ is a mere proposition.*

4.3 Bi-invertible maps

D 4.3.1. We say $f : A \to B$ is **bi-invertible** if it has both a left inverse and a right inverse:

$$\mathsf{biinv}(f) :\equiv \mathsf{linv}(f) \times \mathsf{rinv}(f).$$

T 4.3.2. *For any $f : A \to B$, the type $\mathsf{biinv}(f)$ is a mere proposition.*

C 4.3.3. *For any $f : A \to B$ we have $\mathsf{biinv}(f) \simeq \mathsf{ishae}(f)$.*

4.4 Contractible fibers

D 4.4.1 (Contractible maps). A map $f : A \to B$ is **contractible** if for all $y : B$, the fiber $\mathsf{fib}_f(y)$ is contractible.

T 4.4.2. *For any $f : A \to B$ we have $\mathsf{isContr}(f) \to \mathsf{ishae}(f)$.*

L 4.4.3. *For any f, the type $\mathsf{isContr}(f)$ is a mere proposition.*

T 4.4.4. *For any $f : A \to B$ we have $\mathsf{isContr}(f) \simeq \mathsf{ishae}(f)$.*

C 4.4.5. *If $f : A \to B$ is such that $B \to \mathsf{isequiv}(f)$, then f is an equivalence.*

4.5 On the definition of equivalences

We have shown that all three definitions of equivalence satisfy the three desirable properties and are pairwise equivalent:

$$\mathsf{isContr}(f) \simeq \mathsf{ishae}(f) \simeq \mathsf{biinv}(f).$$

4.6 Surjections and embeddings

When A and B are sets and $f : A \to B$ is an equivalence, we also call it as **isomorphism** or a **bijection**. (We avoid these words for types that are not sets, since in homotopy theory and higher category theory they often denote a stricter notion of "sameness" than homotopy equivalence.) In set theory, a function is a bijection just when it is both injective and surjective. The same is true in type theory, if we formulate these conditions appropriately. For clarity, when dealing with types that are not sets, we will speak of *embeddings* instead of injections.

D 4.6.1. Let $f : A \to B$.

(i). We say f is **surjective** (or a **surjection**) if for every $b : B$ we have $\|\mathsf{fib}_f(b)\|$.

(ii). We say f is an **embedding** if for every $x, y : A$ the function $\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y))$ is an equivalence.

In other words, f is surjective if every fiber of f is merely inhabited, or equivalently if for all $b : B$ there merely exists an $a : A$ such that $f(a) = b$. In traditional logical notation, f is surjective if $\forall(b : B). \exists(a : A). (f(a) = b)$. This must be distinguished from the stronger assertion that $\prod_{(b:B)} \sum_{(a:A)} (f(a) = b)$; if this holds we say that f is a **split surjection**. (Since this latter type is equivalent to $\sum_{(g:B \to A)} \prod_{(b:B)} (f(g(b)) = b)$, being a split surjection is the same as being a *retraction* as defined in **??**.)
The axiom of choice from **??** says exactly that every surjection *between sets* is split. However, in the presence of the univalence axiom, it is simply false that *all* surjections are split. In **??** we constructed a type family $Y : X \to \mathcal{U}$ such that $\prod_{(x:X)} \|Y(x)\|$ but $\neg \prod_{(x:X)} Y(x)$; for any such family, the first projection $(\sum_{(x:X)} Y(x)) \to X$ is a surjection that is not split.
If A and B are sets, then by **??**, f is an embedding just when

$$\prod_{x,y:A} (f(x) =_B f(y)) \to (x =_A y). \qquad (4.6.2)$$

In this case we say that f is **injective**, or an **injection**. We avoid these word for types that are not sets, because they might be interpreted as (4), which is an ill-behaved notion for non-sets. It is also true that any function between sets is surjective if and only if it is an *epimorphism* in a suitable sense, but this also fails for more general types, and surjectivity is generally the more important notion.

T 4.6.3. *A function $f : A \to B$ is an equivalence if and only if it is both surjective and an embedding.*

C 4.6.4. *For any $f : A \to B$ we have*

$$\mathsf{isequiv}(f) \simeq (\mathsf{isEmbedding}(f) \times \mathsf{isSurjective}(f)).$$

4.7 Closure properties of equivalences

T 4.7.1 (The 2-out-of-3 property). *Suppose $f : A \to B$ and $g : B \to C$. If any two of f, g, and $g \circ f$ are equivalences, so is the third.*

D 4.7.2. A function $g : A \to B$ is said to be a **retract** of a function $f : X \to Y$ if there is a diagram

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

for which there are

(i). a homotopy $R : r \circ s \sim \mathsf{id}_A$.

(ii). a homotopy $R' : r' \circ s' \sim \mathsf{id}_B$.

(iii). a homotopy $L : f \circ s \sim s' \circ g$.

(iv). a homotopy $K : g \circ r \sim r' \circ f$.

(v). for every $a : A$, a path $H(a)$ witnessing the commutativity of the square

$$\begin{tikzcd}
g(r(s(a))) \arrow[r, equal, "K(s(a))"] \arrow[d, equal, "g(R(a))"'] & r'(f(s(a))) \arrow[d, equal, "r'(L(a))"] \\
g(a) \arrow[r, equal, "R'(g(a))^{-1}"'] & r'(s'(g(a)))
\end{tikzcd}$$

L 4.7.3. *If a function $g : A \to B$ is a retract of a function $f : X \to Y$, then $\mathsf{fib}_g(b)$ is a retract of $\mathsf{fib}_f(s'(b))$ for every $b : B$, where $s' : B \to Y$ is as in **??**.*

T 4.7.4. *If g is a retract of an equivalence f, then g is also an equivalence.*

D 4.7.5. Given type families $P, Q : A \to \mathcal{U}$ and a map $f : \prod_{(x:A)} P(x) \to Q(x)$, we define

$$\mathsf{total}(f) :\equiv \lambda w. (\mathsf{pr}_1 w, f(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) : \sum_{x:A} P(x) \to \sum_{x:A} Q(x).$$

T 4.7.6. *Suppose that f is a fiberwise transformation between families P and Q over a type A and let $x : A$ and $v : Q(x)$. Then we have an equivalence*

$$\mathsf{fib}_{\mathsf{total}(f)}((x, v)) \simeq \mathsf{fib}_{f(x)}(v).$$

T 4.7.7. *Suppose that f is a fiberwise transformation between families P and Q over a type A. Then f is a fiberwise equivalence if and only if $\mathsf{total}(f)$ is an equivalence.*

4.8 The object classifier

L 4.8.1. *For any type family $B : A \to \mathcal{U}$, the fiber of $\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$ over $a : A$ is equivalent to $B(a)$:*

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

L 4.8.2. *For any function $f : A \to B$, we have $A \simeq \sum_{(b:B)} \mathsf{fib}_f(b)$.*

T 4.8.3. *For any type B there is an equivalence*

$$\chi : \Big(\sum_{A:\mathcal{U}} (A \to B)\Big) \simeq (B \to \mathcal{U}).$$

T 4.8.4. *Let $f : A \to B$ be a function. Then the diagram*

$$\begin{tikzcd}
A \arrow[r, "\vartheta_f"] \arrow[d, "f"'] & \mathcal{U}_\bullet \arrow[d, "\mathsf{pr}_1"] \\
B \arrow[r, "\chi_f"'] & \mathcal{U}
\end{tikzcd}$$

*is a pullback square. (see **??**). Here the function ϑ_f is defined by*

$$\lambda a. (\mathsf{fib}_f(f(a)), (a, \mathsf{refl}_{f(a)})).$$

4.9 Univalence implies function extensionality

D 4.9.1. The **weak function extensionality principle** asserts that there is a function

$$\Big(\prod_{x:A} \mathsf{isContr}(P(x))\Big) \to \mathsf{isContr}\Big(\prod_{x:A} P(x)\Big)$$

for any family $P : A \to \mathcal{U}$ of types over any type A.

L 4.9.2. *Assuming $\mathcal{U}$ is univalent, for any $A, B, X : \mathcal{U}$ and any $e : A \simeq B$, there is an equivalence*

$$(X \to A) \simeq (X \to B)$$

of which the underlying map is given by post-composition with the underlying function of e.

C 4.9.3. *Let $P : A \to \mathcal{U}$ be a family of contractible types, i.e.*

$$\prod_{x:A} \mathsf{isContr}(P(x)).$$

Then the projection $\mathsf{pr}_1 : (\sum_{(x:A)} P(x)) \to A$ is an equivalence. Assuming $\mathcal{U}$ is univalent, it follows immediately that post-composition with pr_1 gives an equivalence

$$\alpha : \Big(A \to \sum_{x:A} P(x)\Big) \simeq (A \to A).$$

T 4.9.4. *In a univalent universe $\mathcal{U}$, suppose that $P : A \to \mathcal{U}$ is a family of contractible types and let α be the function of **??**. Then $\prod_{(x:A)} P(x)$ is a retract of $\mathsf{fib}_\alpha(\mathsf{id}_A)$. As a consequence, $\prod_{(x:A)} P(x)$ is contractible. In other words, the univalence axiom implies the weak function extensionality principle.*

T 4.9.5. *Weak function extensionality implies the function extensionality **??**.*

Exer. 4.45. Consider the type of "two-sided adjoint equivalence data" for $f : A \to B$,

$$\sum_{(g:B\to A)} \sum_{(\eta:g\circ f\sim \mathsf{id}_A)} \sum_{(\epsilon:f\circ g\sim \mathsf{id}_B)} \Big(\prod_{x:A} f(\eta x) = \epsilon(fx)\Big) \times \Big(\prod_{y:B} g(\epsilon y) = \eta(gy)\Big).$$

By **??**, we know that if f is an equivalence, then this type is inhabited. Give a characterization of this type analogous to **??**.
Can you give an example showing that this type is not generally a mere proposition? (This will be easier after **??**.)

Exer. 4.46. Show that for any $A, B : \mathcal{U}$, the following type is equivalent to $A \simeq B$.

$$\sum_{R:A\to B\to \mathcal{U}} \Big(\prod_{a:A} \mathsf{isContr}\Big(\sum_{b:B} R(a,b)\Big)\Big) \times \Big(\prod_{b:B} \mathsf{isContr}\Big(\sum_{a:A} R(a,b)\Big)\Big).$$

Can you extract from this a definition of a type satisfying the three desiderata of $\mathsf{isequiv}(f)$?

Exer. 4.47. Reformulate the proof of **??** without using univalence.

Exer. 4.48 (The unstable octahedral axiom). Suppose $f : A \to B$ and $g : B \to C$ and $b : B$.

(i). Show that there is a natural map $\mathsf{fib}_{g\circ f}(g(b)) \to \mathsf{fib}_g(g(b))$ whose fiber over $(b, \mathsf{refl}_{g(b)})$ is equivalent to $\mathsf{fib}_f(b)$.

(ii). Show that $\mathsf{fib}_{g\circ f}(c) \simeq \sum_{(w:\mathsf{fib}_g(c))} \mathsf{fib}_f(\mathsf{pr}_1 w)$.

Exer. 4.49. Prove that equivalences satisfy the *2-out-of-6 property*: given $f : A \to B$ and $g : B \to C$ and $h : C \to D$, if $g \circ f$ and $h \circ g$ are equivalences, so are f, g, h, and $h \circ g \circ f$. Use this to give a higher-level proof of **??**.

Exer. 4.50. For $A, B : \mathcal{U}$, define

$$\mathsf{idtoqinv}_{A,B} : (A = B) \to \sum_{f:A\to B} \mathsf{qinv}(f)$$

by path induction in the obvious way. Let **qinv-univalence** denote the modified form of the univalence axiom which asserts that for all $A, B : \mathcal{U}$ the function $\mathsf{idtoqinv}_{A,B}$ has a quasi-inverse.

(i). Show that qinv-univalence can be used instead of univalence in the proof of function extensionality in **??**.

(ii). Show that qinv-univalence can be used instead of univalence in the proof of **??**.

(iii). Show that qinv-univalence is inconsistent (i.e. allows construction of an inhabitant of $\mathbf{0}$). Thus, the use of a "good" version of $\mathsf{isequiv}$ is essential in the statement of univalence.

Exer. 4.51. Show that a function $f : A \to B$ is an embedding if and only if the following two conditions hold:

(i). f is *left cancellable*, i.e. for any $x, y : A$, if $f(x) = f(y)$ then $x = y$.

(ii). For any $x : A$, the map $\mathsf{ap}_f : \Omega(A, x) \to \Omega(B, f(x))$ is an equivalence.

(In particular, if A is a set, then f is an embedding if and only if it is left-cancellable and $\Omega(B, f(x))$ is contractible for all $x : A$.) Give examples to show that neither of **??** or **??** implies the other.

Exer. 4.52. Show that the type of left-cancellable functions $\mathbf{2} \to B$ (see **??**) is equivalent to $\sum_{(x,y:B)}(x \neq y)$. Give a similar explicit characterization of the type of embeddings $\mathbf{2} \to B$.

Homotopy Type Theory

Induction

5.1 Introduction to inductive types

T 5.1.1. *Let $f, g : \prod_{(x:\mathbb{N})} E(x)$ be two functions which satisfy the recurrences*

$$e_z : E(0) \quad \textit{and} \quad e_s : \prod_{n:\mathbb{N}} E(n) \to E(\mathsf{succ}(n))$$

up to propositional equality, i.e., such that

$$f(0) = e_z \quad \textit{and} \quad g(0) = e_z$$

as well as

$$\prod_{n:\mathbb{N}} f(\mathsf{succ}(n)) = e_s(n, f(n)),$$
$$\prod_{n:\mathbb{N}} g(\mathsf{succ}(n)) = e_s(n, g(n)).$$

Then f and g are equal.

5.2 Uniqueness of inductive types

5.3 W-types

T 5.3.1. *Let $g, h : \prod_{(w:\mathsf{W}_{(x:A)}B(x))} E(w)$ be two functions which satisfy the recurrence*

$$e : \prod_{a,f} \Big(\prod_{b:B(a)} E(f(b)) \Big) \to E(\mathsf{sup}(a, f)),$$

propositionally, i.e., such that

$$\prod_{a,f} g(\mathsf{sup}(a, f)) = e(a, f, \lambda b.\, g(f(b))),$$
$$\prod_{a,f} h(\mathsf{sup}(a, f)) = e(a, f, \lambda b.\, h(f(b))).$$

Then g and h are equal.

5.4 Inductive types are initial algebras

D 5.4.1. A **$\mathbb{N}$-algebra** is a type C with two elements $c_0 : C$, $c_s : C \to C$. The type of such algebras is

$$\mathbb{N}\mathsf{Alg} :\equiv \sum_{C:\mathcal{U}} C \times (C \to C).$$

D 5.4.2. A **$\mathbb{N}$-homomorphism** between $\mathbb{N}$-algebras (C, c_0, c_s) and (D, d_0, d_s) is a function $h : C \to D$ such that $h(c_0) = d_0$ and $h(c_s(c)) = d_s(h(c))$ for all $c : C$. The type of such homomorphisms is

$$\mathbb{N}\mathsf{Hom}((C, c_0, c_s), (D, d_0, d_s)) :\equiv \sum_{(h:C\to D)} (h(c_0) = d_0) \times \textstyle\prod_{(c:C)} (h(c_s(c)) = d_s(h(c))).$$

D 5.4.3. A $\mathbb{N}$-algebra I is called **homotopy-initial**, or **h-initial** for short, if for any other $\mathbb{N}$-algebra C, the type of $\mathbb{N}$-homomorphisms from I to C is contractible. Thus,

$$\mathsf{isHinit}_{\mathbb{N}}(I) :\equiv \prod_{C:\mathbb{N}\mathsf{Alg}} \mathsf{isContr}(\mathbb{N}\mathsf{Hom}(I, C)).$$

T 5.4.4. *Any two h-initial $\mathbb{N}$-algebras are equal. Thus, the type of h-initial $\mathbb{N}$-algebras is a mere proposition.*

T 5.4.5. *The $\mathbb{N}$-algebra $(\mathbb{N}, \mathbf{0}, \mathsf{succ})$ is homotopy initial.*

T 5.4.6. *For any type $A : \mathcal{U}$ and type family $B : A \to \mathcal{U}$, the W-algebra $(\mathsf{W}_{(x:A)}B(x), \mathsf{sup})$ is h-initial.*

5.5 Homotopy-inductive types

T 5.5.1. *For any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, the type $\mathsf{W}_d(A, B)$ is a mere proposition.*

T 5.5.2. *For any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, the type $\mathsf{W}_s(A, B)$ is a mere proposition.*

T 5.5.3. *For any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, the type $\mathsf{W}_h(A, B)$ is a mere proposition.*

L 5.5.4. *For any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, we have*

$$\mathsf{W}_d(A, B) \simeq \mathsf{W}_s(A, B) \simeq \mathsf{W}_h(A, B)$$

T 5.5.5. *The types satisfying the formation, introduction, elimination, and propositional computation rules for W-types are precisely the homotopy-initial W-algebras.*

T 5.5.6. *The rules for natural numbers with propositional computation rules can be derived from the rules for W-types with propositional computation rules.*

5.6 The general syntax of inductive definitions

NB 5.6.1. There is a question of universe size to be addressed. In general, an inductive type must live in a universe that already contains all the types going into its definition. Thus if in the definition of D, the ambiguous notation Prop means $\mathsf{Prop}_{\mathcal{U}}$, then we do not have $D : \mathcal{U}$ but only $D : \mathcal{U}'$ for some larger universe $\mathcal{U}'$ with $\mathcal{U} : \mathcal{U}'$. In a predicative theory, therefore, the right-hand side of (5) lives in $\mathsf{Prop}_{\mathcal{U}'}$, not $\mathsf{Prop}_{\mathcal{U}}$. So this contradiction does require the propositional resizing axiom mentioned in **??**.

5.7 Generalizations of inductive types

5.8 Identity types and identity systems

D 5.8.1. Let A be a type and $a_0 : A$ an element.

- A **pointed predicate** over (A, a_0) is a family $R : A \to \mathcal{U}$ equipped with an element $r_0 : R(a_0)$.
- For pointed predicates (R, r_0) and (S, s_0), a family of maps $g : \prod_{(b:A)} R(b) \to S(b)$ is **pointed** if $g(a_0, r_0) = s_0$. We have

$$\mathsf{ppmap}(R, S) :\equiv \sum_{g:\prod_{(b:A)} R(b)\to S(b)} (g(a_0, r_0) = s_0).$$

- An **identity system at** a_0 is a pointed predicate (R, r_0) such that for any type family $D : \prod_{(b:A)} R(b) \to \mathcal{U}$ and $d : D(a_0, r_0)$, there exists a function $f : \prod_{(b:A)} \prod_{(r:R(b))} D(b, r)$ such that $f(a_0, r_0) = d$.

T 5.8.2. *For a pointed predicate (R, r_0) over (A, a_0), the following are logically equivalent.*

(i). (R, r_0) *is an identity system at* a_0.
(ii). *For any pointed predicate* (S, s_0), *the type* $\mathsf{ppmap}(R, S)$ *is contractible.*
(iii). *For any* $b : A$, *the function* $\mathsf{transport}^R(-, r_0) : (a_0 =_A b) \to R(b)$ *is an equivalence.*
(iv). *The type* $\sum_{(b:A)} R(b)$ *is contractible.*

D 5.8.3. An **identity system** over a type A is a family $R : A \to A \to \mathcal{U}$ equipped with a function $r_0 : \prod_{(a:A)} R(a, a)$ such that for any type family $D : \prod_{(a,b:A)} R(a, b) \to \mathcal{U}$ and $d : \prod_{(a:A)} D(a, a, r_0(a))$, there exists a function $f : \prod_{(a,b:A)} \prod_{(r:R(a,b))} D(a, b, r)$ such that $f(a, a, r_0(a)) = d(a)$ for all $a : A$.

T 5.8.4. *For $R : A \to A \to \mathcal{U}$ equipped with $r_0 : \prod_{(a:A)} R(a, a)$, the following are logically equivalent.*

(i). (R, r_0) *is an identity system over* A.
(ii). *For all* $a_0 : A$, *the pointed predicate* $(R(a_0), r_0(a_0))$ *is an identity system at* a_0.
(iii). *For any* $S : A \to A \to \mathcal{U}$ *and* $s_0 : \prod_{(a:A)} S(a, a)$, *the type*

$$\sum_{(g:\prod_{(a,b:A)} R(a,b)\to S(a,b))} \prod_{(a:A)} g(a, a, r_0(a)) = s_0(a)$$

is contractible.
(iv). *For any* $a, b : A$, *the map* $\mathsf{transport}^{R(a)}(-, r_0(a)) : (a =_A b) \to R(a, b)$ *is an equivalence.*
(v). *For any* $a : A$, *the type* $\sum_{(b:A)} R(a, b)$ *is contractible.*

C 5.8.5 (Equivalence induction). *Given any type family*

$$D : \prod_{A,B:\mathcal{U}} (A \simeq B) \to \mathcal{U}$$

and function $d : \prod_{(A:\mathcal{U})} D(A, A, \mathsf{id}_A)$, there exists

$$f : \prod_{(A,B:\mathcal{U})} \prod_{(e:A\simeq B)} D(A, B, e)$$

such that $f(A, A, \mathsf{id}_A) = d(A)$ for all $A : \mathcal{U}$.

C 5.8.6 (Homotopy induction). *Given any*

$$D : \prod_{f,g:\prod_{(a:A)} B(a)} (f \sim g) \to \mathcal{U}$$

and $d : \prod_{(f:\prod_{(a:A)} B(a))} D(f, f, \lambda x.\, \mathsf{refl}_{f(x)})$, there exists

$$k : \prod_{(f,g:\prod_{(a:A)} B(a))} \prod_{(h:f\sim g)} D(f, g, h)$$

such that $k(f, f, \lambda x.\, \mathsf{refl}_{f(x)}) = d(f)$ for all f.

Notes

Exercises

ex Derive the induction principle for the type $\mathsf{List}(A)$ of lists from its definition as an inductive type in **??**.

Exer. 5.53. Construct two functions on natural numbers which satisfy the same recurrence (e_z, e_s) judgmentally, but are not judgmentally equal.

Exer. 5.54. Construct two different recurrences (e_z, e_s) on the same type E which are both satisfied judgmentally by the same function $f : \mathbb{N} \to E$.

Exer. 5.55. Show that for any type family $E : \mathbf{2} \to \mathcal{U}$, the induction operator

$$\mathsf{ind}_{\mathbf{2}}(E) : (E(0_{\mathbf{2}}) \times E(1_{\mathbf{2}})) \to \prod_{b:\mathbf{2}} E(b)$$

is an equivalence.

Exer. 5.56. Show that the analogous statement to **??** for $\mathbb{N}$ fails.

Exer. 5.57. Show that if we assume simple instead of dependent elimination for W-types, the uniqueness property (analogue of **??**) fails to hold. That is, exhibit a type satisfying the recursion principle of a W-type, but for which functions are not determined uniquely by their recurrence.

Exer. 5.58. Suppose that in the "inductive definition" of the type C at the beginning of **??**, we replace the type $\mathbb{N}$ by $\mathbf{0}$. Analogously to (5), we might consider a recursion principle for this type with hypothesis

$$h : (C \to \mathbf{0}) \to (P \to \mathbf{0}) \to P.$$

Show that even without a computation rule, this recursion principle is inconsistent, i.e. it allows us to construct an element of $\mathbf{0}$.

Exer. 5.59. Consider now an "inductive type" D with one constructor $\mathsf{scott} : (D \to D) \to D$. The second recursor for C suggested in **??** leads to the following recursor for D:

$$\mathsf{rec}_D : \prod_{P:\mathcal{U}} ((D \to D) \to (D \to P) \to P) \to D \to P$$

with computation rule $\mathsf{rec}_D(P, h, \mathsf{scott}(\alpha)) \equiv h(\alpha, (\lambda d.\, \mathsf{rec}_D(P, h, \alpha(d))))$. Show that this also leads to a contradiction.

Exer. 5.60. Let A be an arbitrary type and consider generally an "inductive definition" of a type L_A with constructor $\mathsf{lawvere} : (L_A \to A) \to L_A$. The second recursor for C suggested in **??** leads to the following recursor for L_A:

$$\mathsf{rec}_{L_A} : \prod_{P:\mathcal{U}} ((L_A \to A) \to P) \to L_A \to P$$

with computation rule $\mathsf{rec}_{L_A}(P, h, \mathsf{lawvere}(\alpha)) \equiv h(\alpha)$. Using this, show that A has the **fixed-point property**, i.e. for every function $f : A \to A$ there exists an $a : A$ such that $f(a) = a$. In particular, L_A is inconsistent if A is a type without the fixed-point property, such as $\mathbf{0}$, $\mathbf{2}$, or $\mathbb{N}$.

Exer. 5.61. Continuing from **??**, consider $L_{\mathbf{1}}$, which is not obviously inconsistent since $\mathbf{1}$ does have the fixed-point property. Formulate an induction principle for $L_{\mathbf{1}}$ and its computation rule, analogously to its recursor, and using this, prove that it is contractible.

Exer. 5.62. In **??** we defined the type $\mathsf{List}(A)$ of finite lists of elements of some type A. Consider a similar inductive definition of a type $\mathsf{Lost}(A)$ whose only constructor is

$$\mathsf{cons} : A \to \mathsf{Lost}(A) \to \mathsf{Lost}(A).$$

Show that $\mathsf{Lost}(A)$ is equivalent to $\mathbf{0}$.

Exer. 5.63. Suppose A is a mere proposition, and $B : A \to \mathcal{U}$.

(i). Show that $\mathsf{W}_{(a:A)} B(a)$ is a mere proposition.

(ii). Show that $\mathsf{W}_{(a:A)} B(a)$ is equivalent to $\sum_{(a:A)} \neg B(a)$.

6.1 Introduction

6.2 Induction principles and dependent paths

NB 6.2.1. Recall that for ordinary inductive types, we regard the computation rules for a recursively defined function as not merely judgmental equalities, but *definitional* ones, and thus we may use the notation $:\equiv$ for them. For instance, the truncated predecessor function $p : \mathbb{N} \to \mathbb{N}$ is defined by $p(0) :\equiv 0$ and $p(\mathsf{succ}(n)) :\equiv n$. In the case of higher inductive types, this sort of notation is reasonable for the point constructors (e.g. $f(\mathsf{base}) :\equiv b$), but for the path constructors it could be misleading, since equalities such as $f(\mathsf{loop}) = \ell$ are not judgmental. Thus, we hybridize the notations, writing instead $f(\mathsf{loop})\ell$ for this sort of "propositional equality by definition".

NB 6.2.2. There are other possible ways to define dependent paths. For instance, instead of $p_*(u) = v$ we could consider $u = (p^{-1})_*(v)$. We could also obtain it as a special case of a more general "heterogeneous equality", or with a direct definition as an inductive type family. All these definitions result in equivalent types, so in that sense it doesn't much matter which we pick. However, choosing $p_*(u) = v$ as the definition makes it easiest to conclude other things about dependent paths, such as the fact that apd_f produces them, or that we can compute them in particular type families using the transport lemmas in **??**.

NB 6.2.3. When describing an application of this induction principle informally, we regard it as a splitting of the goal "$P(x)$ for all $x : \mathbb{S}^1$" into two cases, which we will sometimes introduce with phrases such as "when x is base" and "when x varies along loop", respectively. There is no specific mathematical meaning assigned to "varying along a path": it is just a convenient way to indicate the beginning of the corresponding section of a proof; see **??** for an example.

L 6.2.4. *If A is a type together with $a : A$ and $p : a =_A a$, then there is a function $f : \mathbb{S}^1 \to A$ with*

$$f(\mathsf{base}) :\equiv a$$
$$\mathsf{ap}_f(\mathsf{loop})p.$$

L 6.2.5. *If A is a type and $f, g : \mathbb{S}^1 \to A$ are two maps together with two equalities p, q:*

$$p : f(\mathsf{base}) =_A g(\mathsf{base}),$$
$$q : f(\mathsf{loop}) =_p^{\lambda x.\, x =_A x} g(\mathsf{loop}).$$

Then for all $x : \mathbb{S}^1$ we have $f(x) = g(x)$.

L 6.2.6. *For any type A we have a natural equivalence*

$$(\mathbb{S}^1 \to A) \simeq \sum_{x:A} (x = x).$$

6.3 The interval

L 6.3.1. *The type I is contractible.*

L 6.3.2. *If $f, g : A \to B$ are two functions such that $f(x) = g(x)$ for every $x : A$, then $f = g$ in the type $A \to B$.*

6.4 Circles and spheres

L 6.4.1. $\mathsf{loop} \neq \mathsf{refl}_{\mathsf{base}}$.

L 6.4.2. *There exists an element of $\prod_{(x:\mathbb{S}^1)}(x = x)$ which is not equal to $x \mapsto \mathsf{refl}_x$.*

C 6.4.3. *If the type $\mathbb{S}^1$ belongs to some universe $\mathcal{U}$, then $\mathcal{U}$ is not a 1-type.*

L 6.4.4. *Given $f : A \to B$ and $x, y : A$ and $p, q : x = y$, and $r : p = q$, we have a path $\mathsf{ap}_f^2(r) : f(p) = f(q)$.*

L 6.4.5. *Given $P : A \to \mathcal{U}$ and $x, y : A$ and $p, q : x = y$ and $r : p = q$, for any $u : P(x)$ we have $\mathsf{transport}^2(r, u) : p_*(u) = q_*(u)$.*

L 6.4.6. *Given $P : A \to \mathcal{U}$ and $x, y : A$ and $p, q : x = y$ and $r : p = q$ and a function $f : \prod_{(x:A)} P(x)$, we have $\mathsf{apd}_f^2(r) : \mathsf{apd}_f(p) =_r^P \mathsf{apd}_f(q)$.*

6.5 Suspensions

L 6.5.1. $\Sigma\mathbf{2} \simeq \mathbb{S}^1$.

L 6.5.2. *For a type A and a pointed type (B, b_0), we have*

$$\mathsf{Map}_*(A_+, B) \simeq (A \to B)$$

L 6.5.3. *For pointed types (A, a_0) and (B, b_0) we have*

$$\mathsf{Map}_*(\Sigma A, B) \simeq \mathsf{Map}_*(A, \Omega B).$$

6.6 Cell complexes

6.7 Hubs and spokes

NB 6.7.1. One might question the need for introducing the hub point h; why couldn't we instead simply add paths continuously relating the boundary of the disc to a point *on* that boundary, as shown in **??**? However, this does not work without further modification. For if, given some $f : \mathbb{S}^1 \to X$, we give a path constructor connecting each $f(x)$ to $f(\mathsf{base})$, then what we end up with is more like the picture in **??** of a cone whose vertex is twisted around and glued to some point on its base. The problem is that the specified path from $f(\mathsf{base})$ to itself may not be reflexivity. We could remedy the problem by adding a 2-dimensional path constructor to ensure this, but using a separate hub avoids the need for any path constructors of dimension above 1.

NB 6.7.2. Note also that this "translation" of higher paths into 1-paths does not preserve judgmental computation rules for these paths, though it does preserve propositional ones.

6.8 Pushouts

D 6.8.1. Given a span $\mathscr{D} = (A \xleftarrow{f} C \xrightarrow{g} B)$ and a type D, a **cocone under $\mathscr{D}$ with vertex D** consists of functions $i : A \to D$ and $j : B \to D$ and a homotopy $h : \prod_{(c:C)}(i(f(c)) = j(g(c)))$:

$$\begin{tikzcd}
C \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\
A \arrow[r, "i"'] \arrow[ur, phantom, "\overset{h}{\nearrow}"] & D
\end{tikzcd}$$

We denote by $\mathsf{cocone}_{\mathscr{D}}(D)$ the type of all such cocones, i.e.

$$\mathsf{cocone}_{\mathscr{D}}(D) :\equiv \sum_{(i:A\to D)} \sum_{(j:B\to D)} \prod_{(c:C)} (i(f(c)) = j(g(c))).$$

L 6.8.2. *For any type E, there is an equivalence*

$$(A \sqcup^C B \to E) \;\simeq\; \mathsf{cocone}_{\mathscr{D}}(E).$$

NB 6.8.3. As remarked in **??**, the notations $\wedge$ and $\vee$ for the smash product and wedge of pointed spaces are also used in logic for "and" and "or", respectively. Since types in homotopy type theory can behave either like spaces or like propositions, there is technically a potential for conflict — but since they rarely do both at once, context generally disambiguates. Furthermore, the smash product and wedge only apply to *pointed* spaces, while the only pointed mere proposition is $\top \equiv \mathbf{1}$ — and we have $\mathbf{1} \wedge \mathbf{1} = \mathbf{1}$ and $\mathbf{1} \vee \mathbf{1} = \mathbf{1}$ for either meaning of $\wedge$ and $\vee$.

NB 6.8.4. Note that colimits do not in general preserve truncatedness. For instance, $\mathbb{S}^0$ and $\mathbf{1}$ are both sets, but the pushout of $\mathbf{1} \leftarrow \mathbb{S}^0 \to \mathbf{1}$ is $\mathbb{S}^1$, which is not a set. If we are interested in colimits in the category of n-types, therefore (and, in particular, in the category of sets), we need to "truncate" the colimit somehow. We will return to this point in **??????**.

6.9 Truncations

L 6.9.1. *Suppose given $B : \|A\|_0 \to \mathcal{U}$ together with $g : \prod_{(a:A)} B(|a|_0)$, and assume that each $B(x)$ is a set. Then there exists $f : \prod_{(x:\|A\|_0)} B(x)$ such that $f(|a|_0) \equiv g(a)$ for all $a : A$.*

L 6.9.2. *For any set B and any type A, composition with $|{-}|_0 : A \to \|A\|_0$ determines an equivalence*

$$(\|A\|_0 \to B) \;\simeq\; (A \to B).$$

L 6.9.3. *Let $A \xleftarrow{f} C \xrightarrow{g} B$ be a span of sets. Then for any set E, there is a canonical equivalence*

$$\left(\|A \sqcup^C B\|_0 \to E\right) \;\simeq\; \mathsf{cocone}_{\mathscr{D}}(E).$$

6.10 Quotients

NB 6.10.1. It is not actually necessary for the definition of set-quotients, and most of their properties, that A be a set. However, this is generally the case of most interest.

L 6.10.2. *The function $q : A \to A/R$ is surjective.*

L 6.10.3. *For any set B, precomposing with q yields an equivalence*

$$(A/R \to B) \simeq \Big(\sum_{(f:A\to B)} \prod_{(a,b:A)} R(a,b) \to (f(a) = f(b)) \Big).$$

D 6.10.4. A predicate $P : A \to \mathsf{Prop}$ is an **equivalence class** of a relation $R : A \times A \to \mathsf{Prop}$ if there merely exists an $a : A$ such that for all $b : A$ we have $R(a,b) \simeq P(b)$.

D 6.10.5. We define

$$A /\!\!/ R :\equiv \{ P : A \to \mathsf{Prop} \mid P \text{ is an equivalence class of } R \}.$$

The function $q' : A \to A /\!\!/ R$ is defined by $q'(a) :\equiv P_a$.

T 6.10.6. *For any equivalence relation R on A, the type $A /\!\!/ R$ is equivalent to the set-quotient A/R.*

NB 6.10.7. The previous two constructions provide quotients in generality, but in particular cases there may be easier constructions. For instance, we may define the integers $\mathbb{Z}$ as a set-quotient

$$\mathbb{Z} :\equiv (\mathbb{N} \times \mathbb{N})/\sim$$

where $\sim$ is the equivalence relation defined by

$$(a,b) \sim (c,d) :\equiv (a + d = b + c).$$

In other words, a pair (a,b) represents the integer $a - b$. In this case, however, there are *canonical representatives* of the equivalence classes: those of the form $(n,0)$ or $(0,n)$.

L 6.10.8. *Suppose $\sim$ is a relation on a set A, and there exists an idempotent $r : A \to A$ such that $(r(x) = r(y)) \simeq (x \sim y)$ for all $x, y : A$. (This implies $\sim$ is an equivalence relation.) Then the type*

$$(A/\sim) :\equiv \Big(\sum_{x:A} r(x) = x \Big)$$

satisfies the universal property of the set-quotient of A by $\sim$, and hence is equivalent to it. In other words, there is a map $q : A \to (A/\sim)$ such that for every set B, precomposition with q induces an equivalence

$$\Big((A/\sim) \to B\Big) \simeq \Big(\sum_{(g:A\to B)} \prod_{(x,y:A)} (x \sim y) \to (g(x) = g(y)) \Big). \tag{6.10.9}$$

C 6.10.10. *Suppose $p : A \to B$ is a retraction between sets. Then B is the quotient of A by the equivalence relation $\sim$ defined by*

$$(a_1 \sim a_2) :\equiv (p(a_1) = p(a_2)).$$

NB 6.10.11. **??** applies to $\mathbb{Z}$ with the idempotent $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ defined by

$$r(a,b) = \begin{cases} (a-b, 0) & \text{if } a \geq b, \\ (0, b-a) & \text{otherwise.} \end{cases}$$

(This is a valid definition even constructively, since the relation $\geq$ on $\mathbb{N}$ is decidable.) Thus a non-negative integer is canonically represented as $(k,0)$ and a non-positive one by $(0,m)$, for $k, m : \mathbb{N}$. This division into cases implies the following "induction principle" for integers, which will be useful in **??**. (As usual, we identify a natural number n with the corresponding non-negative integer, i.e. with the image of $(n,0) : \mathbb{N} \times \mathbb{N}$ in $\mathbb{Z}$.)

L 6.10.12. *Suppose $P : \mathbb{Z} \to \mathcal{U}$ is a type family and that we have*

- $d_0 : P(0)$,
- $d_+ : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}(n))$, *and*
- $d_- : \prod_{(n:\mathbb{N})} P(-n) \to P(-\mathsf{succ}(n))$.

Then we have $f : \prod_{(z:\mathbb{Z})} P(z)$ such that

- $f(0) = d_0$,
- $f(\mathsf{succ}(n)) = d_+(n, f(n))$ *for all $n : \mathbb{N}$, and*
- $f(-\mathsf{succ}(n)) = d_-(n, f(-n))$ *for all $n : \mathbb{N}$.*

C 6.10.13. *Let A be a type with $a : A$ and $p : a = a$. There is a function $\prod_{(n:\mathbb{Z})}(a = a)$, denoted $n \mapsto p^n$, defined by*

$$\begin{aligned} & p^0 \mathsf{refl}_a \\ & p^{n+1} p^n \cdot p && \text{for } n \geq 0 \\ & p^{n-1} p^n \cdot p^{-1} && \text{for } n \leq 0. \end{aligned}$$

6.11 Algebra

D 6.11.1. A **monoid** is a set G together with

- a *multiplication* function $G \times G \to G$, written infix as $(x, y) \mapsto x \cdot y$; and
- a *unit* element $e : G$; such that
- for any $x : G$, we have $x \cdot e = x$ and $e \cdot x = x$; and
- for any $x, y, z : G$, we have $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.

A **group** is a monoid G together with

- an *inversion* function $i : G \to G$, written $x \mapsto x^{-1}$; such that
- for any $x : G$ we have $x \cdot x^{-1} = e$ and $x^{-1} \cdot x = e$.

NB 6.11.2. Note that we require a group to be a set. We could consider a more general notion of "∞-group" which is not a set, but this would take us further afield than is appropriate at the moment. With our current definition, we may expect the resulting "group theory" to behave similarly to the way it does in set-theoretic mathematics (with the caveat that, unless we assume LEM, it will be "constructive" group theory).

E 6.11.3. The natural numbers $\mathbb{N}$ are a monoid under addition, with unit 0, and also under multiplication, with unit 1. If we define the arithmetical operations on the integers $\mathbb{Z}$ in the obvious way, then as usual they are a group under addition and a monoid under multiplication (and, of course, a ring). For instance, if $u, v \in \mathbb{Z}$ are represented by (a, b) and (c, d), respectively, then $u + v$ is represented by $(a + c, b + d)$, $-u$ is represented by (b, a), and uv is represented by $(ac + bd, ad + bc)$.

E 6.11.4. We essentially observed in **??** that if (A, a) is a pointed type, then its loop space $\Omega(A, a) :\equiv (a =_A a)$ has all the structure of a group, except that it is not in general a set. It should be an "∞-group" in the sense mentioned in **??**, but we can also make it a group by truncation. Specifically, we define the **fundamental group** of A based at $a : A$ to be

$$\pi_1(A, a) :\equiv \|\Omega(A, a)\|_0.$$

This inherits a group structure; for instance, the multiplication $\pi_1(A, a) \times \pi_1(A, a) \to \pi_1(A, a)$ is defined by double induction on truncation from the concatenation of paths.
More generally, the n^{th} **homotopy group** of (A, a) is $\pi_n(A, a) :\equiv \|\Omega^n(A, a)\|_0$. Then $\pi_n(A, a) = \pi_1(\Omega^{n-1}(A, a))$ for $n \geq 1$, so it is also a group. (When $n = 0$, we have $\pi_0(A) \equiv \|A\|_0$, which is not a group.) Moreover, the Eckmann–Hilton argument (**??**) implies that if $n \geq 2$, then $\pi_n(A, a)$ is an *abelian* group, i.e. we have $x \cdot y = y \cdot x$ for all x, y. **??** will be largely the study of these groups.

L 6.11.5. *For any set A, the type $\mathsf{List}(A)$ is the free monoid on A. In other words, for any monoid G, composition with η is an equivalence*

$$\hom_{\mathrm{Monoid}}(\mathsf{List}(A), G) \simeq (A \to G),$$

where $\hom_{\mathrm{Monoid}}(-, -)$ denotes the set of monoid homomorphisms (functions which preserve the multiplication and unit).

T 6.11.6. *$F(A)$ is the free group on A. In other words, for any (set) group G, composition with $\eta : A \to F(A)$ determines an equivalence*

$$\hom_{\mathrm{Group}}(F(A), G) \simeq (A \to G)$$

where $\hom_{\mathrm{Group}}(-, -)$ denotes the set of group homomorphisms between two groups.

T 6.11.7. *Let A be a set, and let $F'(A)$ be the set-quotient of $\mathsf{List}(A + A)$ by the following relations.*

$$(\dots, a_1, a_2, \widehat{a_2}, a_3, \dots) = (\dots, a_1, a_3, \dots)$$
$$(\dots, a_1, \widehat{a_2}, a_2, a_3, \dots) = (\dots, a_1, a_3, \dots).$$

Then $F'(A)$ is also the free group on the set A.

NB 6.11.8. Nowhere in the construction of $F(A)$ and $F'(A)$, and the proof of their universal properties, did we use the assumption that A is a set. Thus, we can actually construct the free group on an arbitrary type. Comparing universal properties, we conclude that $F(A) \simeq F(\|A\|_0)$.

6.12 The flattening lemma

L 6.12.1. *Given $B : A \to \mathcal{U}$ and $x, y : A$, with a path $p : x = y$ and an equivalence $e : B(x) \simeq B(y)$ such that $B(p) = \mathsf{ua}(e)$, then for any $u : B(x)$ we have*

$$\mathsf{transport}^B(p, u) = e(u).$$

L 6.12.2 (Flattening lemma). *In the above situation, we have*

$$\Big(\sum_{x:W} P(x)\Big) \simeq \widetilde{W}.$$

L 6.12.3. *There are functions*

- $\widetilde{\mathsf{c}}' : \prod_{(a:A)} C(a) \to \sum_{(x:W)} P(x)$ *and*
- $\widetilde{\mathsf{p}}' : \prod_{(b:B)} \prod_{(y:C(f(b)))} \Big(\widetilde{\mathsf{c}}'(f(b),y) =_{\sum_{(w:W)} P(w)} \widetilde{\mathsf{c}}'(g(b), D(b)(y))\Big)$.

L 6.12.4. *Suppose* $Q : (\sum_{(x:W)} P(x)) \to \mathcal{U}$ *is a type family and that we have*

- $c : \prod_{(a:A)} \prod_{(x:C(a))} Q(\widetilde{\mathsf{c}}'(a,x))$ *and*
- $p : \prod_{(b:B)} \prod_{(y:C(f(b)))} \Big(\widetilde{\mathsf{p}}'(b,y)_*(c(f(b),y)) = c(g(b), D(b)(y))\Big)$.

Then there exists $k : \prod_{(z:\sum_{(w:W)} P(w))} Q(z)$ *such that* $k(\widetilde{\mathsf{c}}'(a,x)) \equiv c(a,x)$.

L 6.12.5. *Suppose* Q *is a type and that we have*

- $c : \prod_{(a:A)} C(a) \to Q$ *and*
- $p : \prod_{(b:B)} \prod_{(y:C(f(b)))} \Big(c(f(b),y) =_Q c(g(b), D(b)(y))\Big)$.

Then there exists $k : (\sum_{(w:W)} P(w)) \to Q$ *such that* $k(\widetilde{\mathsf{c}}'(a,x)) \equiv c(a,x)$.

L 6.12.6. *Let* $Y : X \to \mathcal{U}$ *be a type family and let* $k : (\sum_{(x:X)} Y(x)) \to Z$ *be defined componentwise by* $k(x,y) :\equiv d(x)(y)$ *for a curried function* $d : \prod_{(x:X)} Y(x) \to Z$*. Then for any* $s : x_1 =_X x_2$ *and any* $y_1 : Y(x_1)$ *and* $y_2 : Y(x_2)$ *with a path* $r : s_*(y_1) = y_2$*, the path*

$$\mathsf{ap}_k(\mathsf{pair}^{=}(s,r)) : k(x_1,y_1) = k(x_2,y_2)$$

is equal to the composite

$$\begin{aligned}
k(x_1,y_1) &\equiv d(x_1)(y_1) \\
&= \mathsf{transport}^{x\mapsto Z}(s, d(x_1)(y_1)) && \text{(by (??)}^{-1}\text{)} \\
&= \mathsf{transport}^{x\mapsto Z}(s, d(x_1)(s^{-1}{}_*(s_*(y_1)))) \\
&= (\mathsf{transport}^{x\mapsto (Y(x)\to Z)}(s, d(x_1)))(s_*(y_1)) && \text{(by (2))} \\
&= d(x_2)(s_*(y_1)) && \text{(by } \mathsf{happly}(\mathsf{apd}_d(s))(s_*(y_1))\text{)} \\
&= d(x_2)(y_2) && \text{(by } \mathsf{ap}_{d(x_2)}(r)\text{)} \\
&\equiv k(x_2,y_2).
\end{aligned}$$

L 6.12.7. *In the situation of* **??***, we have* $\mathsf{ap}_k(\widetilde{\mathsf{p}}'(b,y)) = p(b,y)$.

6.13 The general syntax of higher inductive definitions

E 6.13.1. Consider a family of functions $f : \prod_{(X:\mathcal{U})} (X \to X)$. Of course, f_X might be just id_X for all X, but other such fs may also exist. For instance, nothing prevents $f_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ from being the nonidentity automorphism (see **??**).
Now suppose that we attempt to define a higher inductive type K generated by:

- two elements $a, b : K$, and
- a path $\sigma : f_K(a) = f_K(b)$.

What would the induction principle for K say? We would assume a type family $P : K \to \mathcal{U}$, and of course we would need $x : P(a)$ and $y : P(b)$. The remaining datum should be a dependent path in P living over σ, which must therefore connect some element of $P(f_K(a))$ to some element of $P(f_K(b))$. But what could these elements possibly be? We know that $P(a)$ and $P(b)$ are inhabited by x and y, respectively, but this tells us nothing about $P(f_K(a))$ and $P(f_K(b))$.

Notes

Exercises

ex Define concatenation of dependent paths, prove that application of dependent functions preserves concatenation, and write out the precise induction principle for the torus T^2 with its computation rules.

Exer. 6.69. Prove that $\Sigma\mathbb{S}^1 \simeq \mathbb{S}^2$, using the explicit definition of $\mathbb{S}^2$ in terms of base and surf given in **??**.

Exer. 6.70. Prove that the torus T^2 as defined in **??** is equivalent to $\mathbb{S}^1 \times \mathbb{S}^1$. (Warning: the path algebra for this is rather difficult.)

Exer. 6.71. Define dependent n-loops and the action of dependent functions on n-loops, and write down the induction principle for the n-spheres as defined at the end of **??**.

Exer. 6.72. Prove that $\Sigma\mathbb{S}^n \simeq \mathbb{S}^{n+1}$, using the definition of $\mathbb{S}^n$ in terms of Ω^n from **??**.

Exer. 6.73. Prove that if the type $\mathbb{S}^2$ belongs to some universe $\mathcal{U}$, then $\mathcal{U}$ is not a 2-type.

Exer. 6.74. Prove that if G is a monoid and $x : G$, then $\sum_{(y:G)}((x \cdot y = e) \times (y \cdot x = e))$ is a mere proposition. Conclude, using the principle of unique choice (**??**), that it would be equivalent to define a group to be a monoid such that for every $x : G$, there merely exists a $y : G$ such that $x \cdot y = e$ and $y \cdot x = e$.

Exer. 6.75. Prove that if A is a set, then $\mathsf{List}(A)$ is a monoid. Then complete the proof of **??**.

Exer. 6.76. Assuming LEM, construct a family $f : \prod_{(X:\mathcal{U})}(X \to X)$ such that $f_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ is the nonidentity automorphism.

Exer. 6.77. Show that the map constructed in **??** is in fact a quasi-inverse to happly, so that an interval type implies the full function extensionality axiom. (You may have to use **??**.)

Exer. 6.78. Prove the universal property of suspension:

$$\Big(\Sigma A \to B\Big) \simeq \Big(\sum_{(b_n:B)} \sum_{(b_s:B)} (A \to (b_n = b_s))\Big)$$

Exer. 6.79. Show that $\mathbb{Z} \simeq \mathbb{N} + \mathbf{1} + \mathbb{N}$. Show that if we were to define $\mathbb{Z}$ as $\mathbb{N} + \mathbf{1} + \mathbb{N}$, then we could obtain **??** with judgmental computation rules.

Exer. 6.80. Show that we can also prove **??** by using $\|\mathbf{2}\|$ instead of I.

Homotopy Type Theory

Homotopy n-types

7.1 Definition of n-types

D 7.1.1. Define the predicate $\mathsf{is}\text{-}n\text{-}\mathsf{type} : \mathcal{U} \to \mathcal{U}$ for $n \geq -2$ by recursion as follows:

$$\mathsf{is}\text{-}n\text{-}\mathsf{type}(X) :\equiv \begin{cases} \mathsf{isContr}(X) & \text{if } n = -2, \\ \prod_{(x,y:X)} \mathsf{is}\text{-}n'\text{-}\mathsf{type}(x =_X y) & \text{if } n = n' + 1. \end{cases}$$

We say that X is an n**-type**, or sometimes that it is n*-truncated*, if $\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ is inhabited.

E 7.1.2. We saw in **??** that X is a (-1)-type if and only if it is a mere proposition. Therefore, X is a 0-type if and only if it is a set.

T 7.1.3. *Let $p : X \to Y$ be a retraction and suppose that X is an n-type, for any $n \geq -2$. Then Y is also an n-type.*

C 7.1.4. *If $X \simeq Y$ and X is an n-type, then so is Y.*

T 7.1.5. *If $f : X \to Y$ is an embedding and Y is an n-type for some $n \geq -1$, then so is X.*

T 7.1.6. *The hierarchy of n-types is cumulative in the following sense: given a number $n \geq -2$, if X is an n-type, then it is also an $(n+1)$-type.*

7.2 Preservation under constructors

T 7.2.1. *Let $n \geq -2$, and let $A : \mathcal{U}$ and $B : A \to \mathcal{U}$. If A is an n-type and for all $a : A$, $B(a)$ is an n-type, then so is $\sum_{(x:A)} B(x)$.*

T 7.2.2. *Let $n \geq -2$, and let $A : \mathcal{U}$ and $B : A \to \mathcal{U}$. If for all $a : A$, $B(a)$ is an n-type, then so is $\prod_{(x:A)} B(x)$.*

T 7.2.3. *For any $n \geq -2$ and any type X, the type $\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ is a mere proposition.*

T 7.2.4. *For any $n \geq -2$, the type $n\text{-}\mathsf{Type}$ is an $(n+1)$-type.*

7.3 Uniqueness of identity proofs and Hedberg's theorem

T 7.3.1. *A type X is a set if and only if it satisfies **Axiom K**: for all $x : X$ and $p : (x =_A x)$ we have $p = \mathsf{refl}_x$.*

T 7.3.2. *Suppose R is a reflexive mere relation on a type X implying identity. Then X is a set, and $R(x,y)$ is equivalent to $x =_X y$ for all $x, y : X$.*

C 7.3.3. *If a type X has the property that $\neg\neg(x = y) \to (x = y)$ for any $x, y : X$, then X is a set.*

L 7.3.4. *For any type A we have $(A + \neg A) \to (\neg\neg A \to A)$.*

T 7.3.5 (Hedberg). *If X has decidable equality, then X is a set.*

T 7.3.6. *The type $\mathbb{N}$ of natural numbers has decidable equality, and hence is a set.*

T 7.3.7. *For any $n \geq -1$, a type X is an $(n+1)$-type if and only if for all $x : X$, the type $\Omega(X, x)$ is an n-type.*

L 7.3.8. *Given $n \geq -1$ and $X : \mathcal{U}$. If, given any inhabitant of X it follows that X is an n-type, then X is an n-type.*

T 7.3.9. *For every $n \geq -1$, a type A is an n-type if and only if $\Omega^{n+1}(A, a)$ is contractible for all $a : A$.*

7.4 Truncations

L 7.4.1. *$\|A\|_n$ is an n-type.*

T 7.4.2. *For any type family $P : \|A\|_n \to \mathcal{U}$ such that each $P(x)$ is an n-type, and any function $g : \prod_{(a:A)} P(|a|_n)$, there exists a section $f : \prod_{(x:\|A\|_n)} P(x)$ such that $f(|a|_n) :\equiv g(a)$ for all $a : A$.*

L 7.4.3 (Universal property of truncations)**.** *Let $n \geq -2$, $A : \mathcal{U}$ and $B : n\text{-}\mathsf{Type}$. The following map is an equivalence:*

$$\begin{cases} (\|A\|_n \to B) & \longrightarrow & (A \to B) \\ g & \longmapsto & g \circ |-|_n \end{cases}$$

L 7.4.4. *Given $f, g : A \to B$ and a homotopy $h : f \sim g$, there is an induced homotopy $\|h\|_n : \|f\|_n \sim \|g\|_n$ such that the composite*

$$|f(a)|_n \xlongequal{\mathsf{nat}^f_n(a)^{-1}} \|f\|_n(|a|_n) \xlongequal{\|h\|_n(|a|_n)} \|g\|_n(|a|_n) \xlongequal{\mathsf{nat}^g_n(a)} |g(a)|_n \tag{7.4.5}$$

is equal to $\mathsf{ap}_{|-|_n}(h(a))$.

C 7.4.6. *A type A is an n-type if and only if $|-|_n : A \to \|A\|_n$ is an equivalence.*

T 7.4.7. *For any types A and B, the induced map $\|A \times B\|_n \to \|A\|_n \times \|B\|_n$ is an equivalence.*

T 7.4.8. *Let $P : A \to \mathcal{U}$ be a family of types. Then there is an equivalence*

$$\Big\|\sum_{x:A} \|P(x)\|_n\Big\|_n \simeq \Big\|\sum_{x:A} P(x)\Big\|_n.$$

C 7.4.9. *If A is an n-type and $P : A \to \mathcal{U}$ is any type family, then*

$$\sum_{a:A} \|P(a)\|_n \simeq \Big\|\sum_{a:A} P(a)\Big\|_n$$

T 7.4.10. *For any A and $x, y : A$ and $n \geq -2$, the map (7) is an equivalence; thus we have*

$$\|x =_A y\|_n \simeq \left(|x|_{n+1} =_{\|A\|_{n+1}} |y|_{n+1}\right).$$

C 7.4.11. *Let $n \geq -2$ and (A, a) be a pointed type. Then*

$$\|\Omega(A, a)\|_n = \Omega(\|(A, a)\|_{n+1})$$

C 7.4.12. *Let $n \geq -2$ and $k \geq 0$ and (A, a) a pointed type. Then*

$$\|\Omega^k(A, a)\|_n = \Omega^k(\|(A, a)\|_{n+k}).$$

L 7.4.13. *Let $k, n \geq -2$ with $k \leq n$ and $A : \mathcal{U}$. Then $\|\|A\|_n\|_k = \|A\|_k$.*

7.5 Colimits of n-types

D 7.5.1. A **span** is a 5-tuple $\mathscr{D} = (A, B, C, f, g)$ with $f : C \to A$ and $g : C \to B$.

$$\mathscr{D} = \begin{tikzcd} C \arrow[r, "g"] \arrow[d, "f"'] & B \\ A & \end{tikzcd}$$

D 7.5.2. Given a span $\mathscr{D} = (A, B, C, f, g)$ and a type D, a **cocone under $\mathscr{D}$ with base D** is a triple (i, j, h) with $i : A \to D$, $j : B \to D$ and $h : \prod_{(c:C)} i(f(c)) = j(g(c))$:

$$\begin{tikzcd} C \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] \arrow[ur, phantom, "\overset{h}{\nearrow}"] & D \end{tikzcd}$$

We denote by $\mathsf{cocone}_{\mathscr{D}}(D)$ the type of all such cocones.

D 7.5.3. Given a span $\mathscr{D}$ of n-types, an n-type D, and a cocone $c : \mathsf{cocone}_{\mathscr{D}}(D)$, the pair (D, c) is said to be a **pushout of $\mathscr{D}$ in n-types** if for every n-type E, the map

$$\begin{cases} (D \to E) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(E) \\ t & \longmapsto & t \circ c \end{cases}$$

is an equivalence.

L 7.5.4. *If (D, c) and (D', c') are two pushouts of $\mathscr{D}$ in $\mathcal{U}_P$, then $(D, c) = (D', c')$.*

C 7.5.5. *The type of pushouts of $\mathscr{D}$ in $\mathcal{U}_P$ is a mere proposition. In particular if pushouts merely exist then they actually exist.*

D 7.5.6. Let

$$\mathscr{D} = \begin{tikzcd} C \arrow[r, "g"] \arrow[d, "f"'] & B \\ A & \end{tikzcd}$$

be a span. We denote by $\bigcirc(\mathscr{D})$ the following span of n-types:

$$\bigcirc(\mathscr{D}) :\equiv \begin{tikzcd} \bigcirc(C) \arrow[r, "\bigcirc(g)"] \arrow[d, "\bigcirc(f)"'] & \bigcirc(B) \\ \bigcirc(A) & \end{tikzcd}$$

D 7.5.7. Let $D : \mathcal{U}$ and $c = (i, j, h) : \mathsf{cocone}_{\mathscr{D}}(D)$. We define

$$\bigcirc(c) = (\bigcirc(i), \bigcirc(j), k) : \mathsf{cocone}_{\bigcirc(\mathscr{D})}(\bigcirc(D))$$

where k is the composite homotopy

$$\bigcirc(i) \circ \bigcirc(f) \sim \bigcirc(i \circ f) \sim \bigcirc(j \circ g) \sim \bigcirc(j) \circ \bigcirc(g)$$

using **??** and the functoriality of $\bigcirc(-)$.

D 7.5.8. Let

$$\mathscr{D} = \begin{tikzcd} C \arrow[r, "g"] \arrow[d, "f"'] & B \\ A & \end{tikzcd} \quad \text{and} \quad \mathscr{D}' = \begin{tikzcd} C' \arrow[r, "g'"] \arrow[d, "f'"'] & B' \\ A' & \end{tikzcd}$$

be spans. A **map of spans** $\mathscr{D} \to \mathscr{D}'$ consists of functions $\alpha : A \to A'$, $\beta : B \to B'$, and $\gamma : C \to C'$ and homotopies $\phi : \alpha \circ f \sim f' \circ \gamma$ and $\psi : \beta \circ g \sim g' \circ \gamma$.

L 7.5.9. *Given $(\alpha, \beta, \gamma, \phi, \psi) : \mathscr{D} \to \mathscr{D}'$ and $t : D \to E$, the following diagram commutes:*

$$\begin{tikzcd} \mathsf{cocone}_{\mathscr{D}'}(D) \arrow[r, "t \circ -"] \arrow[d] & \mathsf{cocone}_{\mathscr{D}'}(E) \arrow[d] \\ \mathsf{cocone}_{\mathscr{D}}(D) \arrow[r, "t \circ -"'] & \mathsf{cocone}_{\mathscr{D}}(E) \end{tikzcd}$$

T 7.5.10. *Let $\mathscr{D}$ be a span and (D, c) its pushout. Then $(\|D\|_n, \|c\|_n)$ is a pushout of $\|\mathscr{D}\|_n$ in n-types.*

7.6 Connectedness

D 7.6.1. A function $f : A \to B$ is said to be n**-connected** if for all $b : B$, the type $\left\|\mathsf{fib}_f(b)\right\|_n$ is contractible:

$$\mathsf{conn}_n(f) :\equiv \prod_{b:B} \mathsf{isContr}(\left\|\mathsf{fib}_f(b)\right\|_n).$$

A type A is said to be n**-connected** if the unique function $A \to \mathbf{1}$ is n-connected, i.e. if $\|A\|_n$ is contractible.

L 7.6.2. *A function f is (-1)-connected if and only if it is surjective in the sense of* **??**.

NB 7.6.3. While our notion of n-connectedness for types agrees with the standard notion in homotopy theory, our notion of n-connectedness for *functions* is off by one from a common indexing in classical homotopy theory. Whereas we say a function f is n-connected if all its fibers are n-connected, some classical homotopy theorists would call such a function $(n+1)$-connected. (This is due to a historical focus on *cofibers* rather than fibers.)

L 7.6.4. *Suppose that g is a retract of a n-connected function f. Then g is n-connected.*

C 7.6.5. *If g is homotopic to a n-connected function f, then g is n-connected.*

L 7.6.6. *Suppose that $f : A \to B$ is n-connected. Then $g : B \to C$ is n-connected if and only if $g \circ f$ is n-connected.*

L 7.6.7. *For $f : A \to B$ and $P : B \to \mathcal{U}$, consider the following function:*

$$\lambda s.\, s \circ f : \Big(\prod_{b:B} P(b)\Big) \to \Big(\prod_{a:A} P(f(a))\Big).$$

For a fixed f and $n \geq -2$, the following are equivalent.

(i). *f is n-connected.*
(ii). *For every $P : B \to n\text{-}\mathsf{Type}$, the map $\lambda s.\, s \circ f$ is an equivalence.*
(iii). *For every $P : B \to n\text{-}\mathsf{Type}$, the map $\lambda s.\, s \circ f$ has a section.*

C 7.6.8. *For any A, the canonical function $|{-}|_n : A \to \|A\|_n$ is n-connected.*

C 7.6.9. *A type A is n-connected if and only if the map*

$$\lambda b.\, \lambda a.\, b : B \to (A \to B)$$

is an equivalence for every n-type B. In other words, "every map from A to an n-type is constant".

L 7.6.10. *Let B be an n-type and let $f : A \to B$ be a function. Then the induced function $g : \|A\|_n \to B$ is an equivalence if and only if f is n-connected.*

L 7.6.11. *Let A be a type and $a_0 : \mathbf{1} \to A$ a basepoint, with $n \geq -1$. Then A is n-connected if and only if the map a_0 is $(n-1)$-connected.*

L 7.6.12. *Let $f : A \to B$ be a function and $P : A \to \mathcal{U}$ and $Q : B \to \mathcal{U}$ be type families. Suppose that $g : \prod_{(a:A)} P(a) \to Q(f(a))$ is a fiberwise n-connected family of functions, i.e. each function $g_a : P(a) \to Q(f(a))$ is n-connected. If f is also n-connected, then so is the function*

$$\varphi : \Big(\sum_{a:A} P(a)\Big) \to \Big(\sum_{b:B} Q(b)\Big)$$
$$\varphi(a,u) :\equiv (f(a), g_a(u)).$$

Conversely, if φ and each g_a are n-connected, and moreover Q is fiberwise merely inhabited (i.e. we have $\|Q(b)\|$ for all $b : B$), then f is n-connected.

L 7.6.13. *Let $P, Q : A \to \mathcal{U}$ be type families and consider a fiberwise transformation*

$$f : \prod_{a:A} \Big(P(a) \to Q(a)\Big)$$

from P to Q. Then the induced map $\mathsf{total}(f) : \sum_{(a:A)} P(a) \to \sum_{(a:A)} Q(a)$ is n-connected if and only if each $f(a)$ is n-connected.

L 7.6.14. *If $f : A \to B$ is n-connected, then it induces an equivalence $\|A\|_n \simeq \|B\|_n$.*

7.7 Orthogonal factorization

D 7.7.1. A function $f : A \to B$ is n**-truncated** if the fiber $\mathsf{fib}_f(b)$ is an n-type for all $b : B$.

L 7.7.2. *For any $n \geq -2$, a function $f : A \to B$ is $(n+1)$-truncated if and only if for all $x, y : A$, the map $\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$ is n-truncated. In particular, f is (-1)-truncated if and only if it is an embedding in the sense of* **??**.

D 7.7.3. Let $f : A \to B$ be a function. The n**-image** of f is defined as

$$\mathsf{im}_n(f) :\equiv \sum_{b:B} \left\| \mathsf{fib}_f(b) \right\|_n.$$

When $n = -1$, we write simply $\mathsf{im}(f)$ and call it the **image** of f.

L 7.7.4. *For any function $f : A \to B$, the canonical function $\tilde{f} : A \to \mathsf{im}_n(f)$ is n-connected. Consequently, any function factors as an n-connected function followed by an n-truncated function.*

L 7.7.5. *Suppose we have a commutative diagram of functions*

$$\begin{tikzcd}
A \arrow[r, "g_1"] \arrow[d, "g_2"'] & X_1 \arrow[d, "h_1"] \\
X_2 \arrow[r, "h_2"'] & B
\end{tikzcd}$$

with $H : h_1 \circ g_1 \sim h_2 \circ g_2$, where g_1 and g_2 are n-connected and where h_1 and h_2 are n-truncated. Then there is an equivalence

$$E(H, b) : \mathsf{fib}_{h_1}(b) \simeq \mathsf{fib}_{h_2}(b)$$

for any $b : B$, such that for any $a : A$ we have an identification

$$\overline{E}(H, a) : E(H, h_1(g_1(a)))(g_1(a), \mathsf{refl}_{h_1(g_1(a))}) = (g_2(a), H(a)^{-1}).$$

T 7.7.6. *For each $f : A \to B$, the space ${}_n(f)$ defined by*

$$\sum_{(X:\mathcal{U})} \sum_{(g:A\to X)} \sum_{(h:X\to B)} (h \circ g \sim f) \times \mathsf{conn}_n(g) \times \mathsf{trunc}_n(h)$$

is contractible. Its center of contraction is the element

$$(\mathsf{im}_n(f), \tilde{f}, \mathsf{pr}_1, \theta, \varphi, \psi) :_n (f)$$

arising from **??**, *where $\theta : \mathsf{pr}_1 \circ \tilde{f} \sim f$ is the canonical homotopy, where φ is the proof of* **??**, *and where ψ is the obvious proof that $\mathsf{pr}_1 : \mathsf{im}_n(f) \to B$ has n-truncated fibers.*

T 7.7.7. *Let $e : A \to B$ be n-connected and $m : C \to D$ be n-truncated. Then the map*

$$\varphi : (B \to C) \ \to \sum_{(h:A\to C)} \sum_{(k:B\to D)} (m \circ h \sim k \circ e)$$

is an equivalence.

L 7.7.8. *Suppose that the square*

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

is a pullback square and let $b : B$. Then $\mathsf{fib}_f(b) \simeq \mathsf{fib}_g(h(b))$.

T 7.7.9. *Consider functions $f : A \to B$, $g : C \to D$ and the diagram*

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "\tilde{f}_n"'] & C \arrow[d, "\tilde{g}_n"] \\
\mathsf{im}_n(f) \arrow[r] \arrow[d, "\mathsf{pr}_1"'] & \mathsf{im}_n(g) \arrow[d, "\mathsf{pr}_1"] \\
B \arrow[r, "h"'] & D
\end{tikzcd}$$

If the outer rectangle is a pullback, then so is the bottom square (and hence so is the top square, by **??***). Consequently, images are stable under pullbacks.*

7.8 Modalities

D 7.8.1. A **reflective subuniverse** is a predicate $P : \mathcal{U} \to \mathsf{Prop}$ such that for every $A : \mathcal{U}$ we have a type $\bigcirc A$ such that $P(\bigcirc A)$ and a map $\eta_A : A \to \bigcirc A$, with the property that for every $B : \mathcal{U}$ with $P(B)$, the following map is an equivalence:

$$\begin{cases} (\bigcirc A \to B) & \longrightarrow & (A \to B) \\ f & \longmapsto & f \circ \eta_A \end{cases}.$$

T 7.8.2. *If $B : A \to \mathcal{U}_P$ is any family of types in a reflective subuniverse $\mathcal{U}_P$, then $\prod_{(x:A)} B(x)$ is also in $\mathcal{U}_P$.*

C 7.8.3. *For any types A and B and any reflective subuniverse, the induced map $\bigcirc(A \times B) \to \bigcirc(A) \times \bigcirc(B)$ is an equivalence.*

T 7.8.4. *For a reflective subuniverse $\mathcal{U}_P$, the following are logically equivalent.*

(i). *If $A : \mathcal{U}_P$ and $B : A \to \mathcal{U}_P$, then $\sum_{(x:A)} B(x)$ is in $\mathcal{U}_P$.*

(ii). *for every $A : \mathcal{U}$, type family $B : \bigcirc A \to \mathcal{U}_P$, and map $g : \prod_{(a:A)} B(\eta(a))$, there exists $f : \prod_{(z:\bigcirc A)} B(z)$ such that $f(\eta(a)) = g(a)$ for all $a : A$.*

D 7.8.5. A **modality** is an operation $\bigcirc : \mathcal{U} \to \mathcal{U}$ for which there are

(i). functions $\eta^{\bigcirc}_A : A \to \bigcirc(A)$ for every type A.

(ii). for every $A : \mathcal{U}$ and every type family $B : \bigcirc(A) \to \mathcal{U}$, a function

$$\mathsf{ind}_{\bigcirc} : \Big(\prod_{a:A} \bigcirc(B(\eta^{\bigcirc}_A(a)))\Big) \to \prod_{z:\bigcirc(A)} \bigcirc(B(z)).$$

(iii). A path $\mathsf{ind}_{\bigcirc}(f)(\eta^{\bigcirc}_A(a)) = f(a)$ for each $f : \prod_{(a:A)} \bigcirc(B(\eta^{\bigcirc}_A(a)))$.

(iv). For any $z, z' : \bigcirc(A)$, the function $\eta^{\bigcirc}_{z=z'} : (z = z') \to \bigcirc(z = z')$ is an equivalence.

We say that A is **modal** for $\bigcirc$ if $\eta^{\bigcirc}_A : A \to \bigcirc(A)$ is an equivalence, and we write

$$\mathcal{U}_{\bigcirc} :\equiv \{\, X : \mathcal{U} \mid X \text{ is } \bigcirc\text{-modal} \,\} \tag{7.8.6}$$

for the type of modal types.

T 7.8.7. *Let A be a type and let $B : \bigcirc(A) \to \mathcal{U}_{\bigcirc}$. Then the function*

$$(- \circ \eta^{\bigcirc}_A) : \Big(\prod_{z:\bigcirc(A)} B(z)\Big) \to \Big(\prod_{a:A} B(\eta^{\bigcirc}_A(a))\Big)$$

is an equivalence.

C 7.8.8. *For any modality $\bigcirc$, the $\bigcirc$-modal types form a reflective subuniverse satisfying the equivalent conditions of* **??***.*

Notes

Exercises

document

Homotopy Type Theory

Homotopy theory

D 8.0.9 (Homotopy Groups). Given $n \geq 1$ and (A, a) a pointed type, we define the **homotopy groups** of A at a by

$$\pi_n(A, a) :\equiv \left\| \Omega^n(A, a) \right\|_0$$

8.1 $\pi_1(S^1)$

8.1.1 Getting started

8.1.2 The classical proof

8.1.3 The universal cover in type theory

D 8.1.1 (Universal Cover of $\mathbb{S}^1$). Define $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ by circle-recursion, with

$$\mathsf{code}(\mathsf{base}) :\equiv \mathbb{Z}$$
$$\mathsf{ap}_{\mathsf{code}}(\mathsf{loop})\mathsf{ua}(\mathsf{succ}).$$

L 8.1.2. $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, x) = x + 1$ *and* $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}, x) = x - 1$.

8.1.4 The encode-decode proof

D 8.1.3. Define $\mathsf{encode} : \prod_{(x:\mathbb{S}^1)} (\mathsf{base} = x) \to \mathsf{code}(x)$ by

$$\mathsf{encode}\; p :\equiv \mathsf{transport}^{\mathsf{code}}(p, 0)$$

(we leave the argument x implicit).

D 8.1.4. Define $\mathsf{decode} : \prod_{(x:\mathbb{S}^1)} \mathsf{code}(x) \to (\mathsf{base} = x)$ by circle induction on x. It suffices to give a function $\mathsf{code}(\mathsf{base}) \to (\mathsf{base} = \mathsf{base})$, for which we use loop^{-}, and to show that loop^{-} respects the loop.

L 8.1.5. *For all* $x : \mathbb{S}^1$ *and* $p : \mathsf{base} = x$, $\mathsf{decode}_x(\mathsf{encode}_x(p)) = p$.

L 8.1.6. *For all* $x : \mathbb{S}^1$ *and* $c : \mathsf{code}(x)$, *we have* $\mathsf{encode}_x(\mathsf{decode}_x(c)) = c$.

T 8.1.7. *There is a family of equivalences* $\prod_{(x:\mathbb{S}^1)}((\mathsf{base} = x) \simeq \mathsf{code}(x))$.

C 8.1.8. $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$.

C 8.1.9. $\pi_1(\mathbb{S}^1) = \mathbb{Z}$, *while* $\pi_n(\mathbb{S}^1) = 0$ *for* $n > 1$.

8.1.5 The homotopy-theoretic proof

L 8.1.10. *The type* $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ *is contractible.*

C 8.1.11. *The map induced by* encode*:*

$$\sum_{(x:\mathbb{S}^1)} (\mathsf{base} = x) \to \sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$$

is an equivalence.

T 8.1.12. $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$.

8.1.6 The universal cover as an identity system

T 8.1.13. *The pair* $(\mathsf{code}, 0)$ *is an identity system at* $\mathsf{base} : \mathbb{S}^1$ *in the sense of* **??**.

C 8.1.14. *For any* $x : \mathbb{S}^1$*, we have* $(\mathsf{base} = x) \simeq \mathsf{code}(x)$.

NB 8.1.15. Note that all of the above proofs that $\pi_1(\mathbb{S}^1) \simeq \mathbb{Z}$ use the univalence axiom in an essential way. This is unavoidable: univalence or something like it is *necessary* in order to prove $\pi_1(\mathbb{S}^1) \simeq \mathbb{Z}$. In the absence of univalence, it is consistent to assume the statement "all types are sets" (a.k.a. "uniqueness of identity proofs" or "Axiom K", as discussed in **??**), and this statement implies instead that $\pi_1(\mathbb{S}^1) \simeq \mathbf{1}$. In fact, the (non)triviality of $\pi_1(\mathbb{S}^1)$ detects exactly whether all types are sets: the proof of **??** showed conversely that if $\mathsf{loop} = \mathsf{refl}_{\mathsf{base}}$ then all types are sets.

8.2 Connectedness of suspensions

T 8.2.1. *If* A *is* n*-connected then the suspension of* A *is* $(n+1)$*-connected.*

C 8.2.2. *For all* $n : \mathbb{N}$*, the sphere* $\mathbb{S}^n$ *is* $(n-1)$*-connected.*

8.3 $\pi_{k \le n}$ of an n-connected space and $\pi_{k<n}(\mathbb{S}^n)$

L 8.3.1. *If* A *is* n*-truncated and* $a : A$*, then* $\pi_k(A, a) = \mathbf{1}$ *for all* $k > n$.

L 8.3.2. *If* A *is* n*-connected and* $a : A$*, then* $\pi_k(A, a) = \mathbf{1}$ *for all* $k \le n$.

C 8.3.3. $\pi_k(\mathbb{S}^n) = \mathbf{1}$ *for* $k < n$.

8.4 Fiber sequences and the long exact sequence

D 8.4.1. A **pointed map** between pointed types (X, x_0) and (Y, y_0) is a map $f : X \to Y$ together with a path $f_0 : f(x_0) = y_0$.

D 8.4.2. Given a pointed map between pointed types $f : X \to Y$, we define a pointed map $\Omega f : \Omega X \to \Omega Y$ by

$$(\Omega f)(p) :\equiv {f_0}^{-1} \cdot f(p) \cdot f_0.$$

The path $(\Omega f)_0 : (\Omega f)(\mathsf{refl}_{x_0}) = \mathsf{refl}_{y_0}$, which exhibits Ωf as a pointed map, is the obvious path of type

$${f_0}^{-1} \cdot f(\mathsf{refl}_{x_0}) \cdot f_0 = \mathsf{refl}_{y_0}.$$

D 8.4.3. The **fiber sequence** of a pointed map $f : X \to Y$ is the infinite sequence of pointed types and pointed maps

$$\begin{tikzcd} \dots \arrow[r, "f^{(n+1)}"] & X^{(n+1)} \arrow[r, "f^{(n)}"] & X^{(n)} \arrow[r, "f^{(n-1)}"] & \dots \arrow[r] & X^{(2)} \arrow[r, "f^{(1)}"] & X^{(1)} \arrow[r, "f^{(0)}"] & X^{(0)} \end{tikzcd}$$

defined recursively by

$$X^{(0)} :\equiv Y \qquad X^{(1)} :\equiv X \qquad f^{(0)} :\equiv f$$

and

$$\begin{aligned} X^{(n+1)} &:\equiv \mathsf{fib}_{f^{(n-1)}}(x_0^{(n-1)}) \\ f^{(n)} &:\equiv \mathsf{pr}_1 \qquad : X^{(n+1)} \to X^{(n)}. \end{aligned}$$

where $x_0^{(n)}$ denotes the basepoint of $X^{(n)}$, chosen recursively as above.

L 8.4.4. *Let* $f : X \to Y$ *be a pointed map of pointed spaces. Then:*

(i). *The fiber of* $f^{(1)} :\equiv \mathsf{pr}_1 : \mathsf{fib}_f(y_0) \to X$ *is equivalent to* ΩY.

(ii). *Similarly, the fiber of* $f^{(2)} : \Omega Y \to \mathsf{fib}_f(y_0)$ *is equivalent to* ΩX.

(iii). *Under these equivalences, the pointed map* $f^{(3)} : \Omega X \to \Omega Y$ *is identified with the pointed map* $\Omega f \circ (-)^{-1}$.

D 8.4.5. An **exact sequence of pointed sets** is a (possibly bounded) sequence of pointed sets and pointed maps:

$$\begin{tikzcd} \dots \arrow[r] & A^{(n+1)} \arrow[r, "f^{(n)}"] & A^{(n)} \arrow[r, "f^{(n-1)}"] & A^{(n-1)} \arrow[r] & \dots \end{tikzcd}$$

such that for every n, the image of $f^{(n)}$ is equal, as a subset of $A^{(n)}$, to the kernel of $f^{(n-1)}$. In other words, for all $a : A^{(n)}$ we have

$$(f^{(n-1)}(a) = a_0^{(n-1)}) \iff \exists (b : A^{(n+1)}).\,(f^{(n)}(b) = a).$$

where $a_0^{(n)}$ denotes the basepoint of $A^{(n)}$.

T 8.4.6. *Let $f : X \to Y$ be a pointed map between pointed spaces with fiber $F :\equiv \mathsf{fib}_f(y_0)$. Then we have the following long exact sequence, which consists of groups except for the last three terms, and abelian groups except for the last six.*

$$\begin{tikzcd} \vdots & \vdots & \vdots \arrow[dll] \\ \pi_k(F) \arrow[r] & \pi_k(X) \arrow[r] & \pi_k(Y) \arrow[dll] \\ \vdots & \vdots & \vdots \arrow[dll] \\ \pi_2(F) \arrow[r] & \pi_2(X) \arrow[r] & \pi_2(Y) \arrow[dll] \\ \pi_1(F) \arrow[r] & \pi_1(X) \arrow[r] & \pi_1(Y) \arrow[dll] \\ \pi_0(F) \arrow[r] & \pi_0(X) \arrow[r] & \pi_0(Y) \end{tikzcd}$$

L 8.4.7. *Suppose given an exact sequence of abelian groups:*

$$\begin{tikzcd} K \arrow[r] & G \arrow[r, "f"] & H \arrow[r] & Q. \end{tikzcd}$$

(i). If $K = 0$, then f is injective.
(ii). If $Q = 0$, then f is surjective.
(iii). If $K = Q = 0$, then f is an isomorphism.

C 8.4.8. *Let $f : A \to B$ be n-connected and $a : A$, and define $b :\equiv f(a)$. Then:*
(i). If $k \leq n$, then $\pi_k(f) : \pi_k(A, a) \to \pi_k(B, b)$ is an isomorphism.
(ii). If $k = n + 1$, then $\pi_k(f) : \pi_k(A, a) \to \pi_k(B, b)$ is surjective.

8.5 The Hopf fibration

T 8.5.1 (Hopf Fibration). *There is a fibration H over $\mathbb{S}^2$ whose fiber over the basepoint is $\mathbb{S}^1$ and whose total space is $\mathbb{S}^3$.*

C 8.5.2. *We have $\pi_2(\mathbb{S}^2) \simeq \mathbb{Z}$ and $\pi_k(\mathbb{S}^3) \simeq \pi_k(\mathbb{S}^2)$ for every $k \geq 3$ (where the map is induced by the Hopf fibration, seen as a map from the total space $\mathbb{S}^3$ to the base space $\mathbb{S}^2$).*

8.5.1 Fibrations over pushouts

L 8.5.3. *Let $\mathscr{D} = (Y \xleftarrow{j} X \xrightarrow{k} Z)$ be a span and assume that we have*

- *Two fibrations $E_Y : Y \to \mathcal{U}$ and $E_Z : Z \to \mathcal{U}$.*
- *An equivalence e_X between $E_Y \circ j : X \to \mathcal{U}$ and $E_Z \circ k : X \to \mathcal{U}$, i.e.*

$$e_X : \prod_{x:X} E_Y(j(x)) \simeq E_Z(k(x)).$$

Then we can construct a fibration $E : Y \sqcup^X Z \to \mathcal{U}$ such that

- *For all $y : Y$, $E(\mathsf{inl}(y)) \equiv E_Y(y)$.*
- *For all $z : Z$, $E(\mathsf{inr}(z)) \equiv E_Z(z)$.*
- *For all $x : X$, $E(\mathsf{glue}(x)) = \mathsf{ua}(e_X(x))$ (note that both sides of the equation are paths in $\mathcal{U}$ from $E_Y(j(x))$ to $E_Z(k(x))$).*

Moreover, the total space of this fibration fits in the following pushout square:

$$\begin{tikzcd}
\sum_{(x:X)} E_Y(j(x)) \arrow[r, "\mathsf{id}\times e_X", "\sim"'] \arrow[d, "j\times\mathsf{id}"'] & \sum_{(x:X)} E_Z(k(x)) \arrow[r, "k\times\mathsf{id}"] & \sum_{(z:Z)} E_Z(z) \arrow[d, "\mathsf{inr}"] \\
\sum_{(y:Y)} E_Y(y) \arrow[rr, "\mathsf{inl}"'] & & \sum_{(t:Y\sqcup^X Z)} E(t)
\end{tikzcd}$$

8.5.2 The Hopf construction

D 8.5.4. An **H-space** consists of

- a type A,
- a base point $e : A$,
- a binary operation $\mu : A \times A \to A$, and
- for every $a : A$, equalities $\mu(e, a) = a$ and $\mu(a, e) = a$.

L 8.5.5. *Let A be a connected H-space. Then for every $a : A$, the maps $\mu(a, -) : A \to A$ and $\mu(-, a) : A \to A$ are equivalences.*

D 8.5.6. Let A be a connected H-space. We define a fibration over ΣA using **??**.
Given that ΣA is the pushout $\mathbf{1} \sqcup^A \mathbf{1}$, we can define a fibration over ΣA by specifying

- two fibrations over $\mathbf{1}$ (i.e. two types F_1 and F_2), and
- a family $e : A \to (F_1 \simeq F_2)$ of equivalences between F_1 and F_2, one for every element of A.

We take A for F_1 and F_2, and for $a : A$ we take the equivalence $\mu(a, -)$ for $e(a)$.

L 8.5.7. *Given a connected H-space A, there is a fibration, called the **Hopf construction**, over ΣA with fiber A and total space $A * A$.*

8.5.3 The Hopf fibration

L 8.5.8. *There is an H-space structure on the circle $\mathbb{S}^1$.*

L 8.5.9. *The operation of join is associative: if A, B and C are three types then we have an equivalence $(A * B) * C \simeq A * (B * C)$.*

L 8.5.10. *For any type A, there is an equivalence $\Sigma A \simeq \mathbf{2} * A$.*

T 8.5.11. *There is a fibration over $\mathbb{S}^2$ of fiber $\mathbb{S}^1$ and total space $\mathbb{S}^3$.*

8.6 The Freudenthal suspension theorem

L 8.6.1. *If $f : A \to B$ is n-connected and $P : B \to k\text{-}\mathsf{Type}$ is a family of k-types for $k \geq n$, then the induced function*

$$(- \circ f) : \Big(\prod_{b:B} P(b)\Big) \to \Big(\prod_{a:A} P(f(a))\Big)$$

is $(k - n - 2)$-truncated.

L 8.6.2 (Wedge connectivity lemma). *Suppose that (A, a_0) and (B, b_0) are n- and m-connected pointed types, respectively, with $n, m \geq 0$, and let*

$$P : A \to B \to (n+m)\text{-}\mathsf{Type}.$$

Then for any $f : \prod_{(a:A)} P(a, b_0)$ and $g : \prod_{(b:B)} P(a_0, b)$ with $p : f(a_0) = g(b_0)$, there exists $h : \prod_{(a:A)} \prod_{(b:B)} P(a, b)$ with homotopies

$$q : \prod_{a:A} h(a, b_0) = f(a) \qquad \text{and} \qquad r : \prod_{b:B} h(a_0, b) = g(b)$$

such that $p = q(a_0)^{-1} \cdot r(b_0)$.

NB 8.6.3. In classical algebraic topology, one considers the *reduced suspension*, in which the path $\mathsf{merid}(x_0)$ is collapsed down to a point, identifying N and S. The reduced and unreduced suspensions are homotopy equivalent, so the distinction is invisible to our purely homotopy-theoretic eyes — and higher inductive types only allow us to "identify" points up to a higher path anyway, there is no purpose to considering reduced suspensions in homotopy type theory. However, the "unreducedness" of our suspension is the reason for the (possibly unexpected) appearance of $\mathsf{merid}(x_0)^{-1}$ in the definition of σ.

T 8.6.4 (The Freudenthal suspension theorem). *Suppose that X is n-connected and pointed, with $n \geq 0$. Then the map $\sigma : X \to \Omega\Sigma(X)$ is $2n$-connected.*

D 8.6.5. If X is n-connected and pointed with $n \geq 0$, then there is a family

$$\mathsf{code} : \prod_{y:\Sigma X} (\mathsf{N} = y) \to \mathcal{U} \tag{8.6.6}$$

such that

$$\mathsf{code}(\mathsf{N}, p) :\equiv \|\mathsf{fib}_\sigma(p)\|_{2n} \equiv \left\|\textstyle\sum_{(x:X)} (\mathsf{merid}(x) \cdot \mathsf{merid}(x_0)^{-1} = p)\right\|_{2n} \tag{8.6.7}$$

$$\mathsf{code}(\mathsf{S}, q) :\equiv \|\mathsf{fib}_{\mathsf{merid}}(q)\|_{2n} \equiv \left\|\textstyle\sum_{(x:X)} (\mathsf{merid}(x) = q)\right\|_{2n}. \tag{8.6.8}$$

L 8.6.9. *Let $A : \mathcal{U}$, $B : A \to \mathcal{U}$, and $C : \prod_{(a:A)} B(a) \to \mathcal{U}$, and also $a_1, a_2 : A$ with $m : a_1 = a_2$ and $b : B(a_2)$. Then the function*

$$\mathsf{transport}^{\widehat{C}}(\mathsf{pair}^=(m, t), -) : C(a_1, \mathsf{transport}^B(m^{-1}, b)) \to C(a_2, b),$$

where $t : \mathsf{transport}^B(m, \mathsf{transport}^B(m^{-1}, b)) = b$ is the obvious coherence path and $\widehat{C} : (\sum_{(a:A)} B(a)) \to \mathcal{U}$ is the uncurried form of C, is equal to the equivalence obtained by univalence from the composite

$$\begin{aligned} C(a_1, \mathsf{transport}^B(m^{-1}, b)) &= \mathsf{transport}^{\lambda a.\, B(a) \to \mathcal{U}}(m, C(a_1))(b) && \text{(by (2))} \\ &= C(a_2, b). && \text{(by } \mathsf{happly}(\mathsf{apd}_C(m), b)\text{)} \end{aligned}$$

C 8.6.10 (Freudenthal Equivalence). *Suppose that X is n-connected and pointed, with $n \geq 0$. Then $\|X\|_{2n} \simeq \|\Omega\Sigma(X)\|_{2n}$.*

C 8.6.11 (Stability for Spheres). *If $k \leq 2n - 2$, then $\pi_{k+1}(S^{n+1}) = \pi_k(S^n)$.*

T 8.6.12. *$\pi_n(\mathbb{S}^n) = \mathbb{Z}$ for every $n \geq 1$.*

C 8.6.13. *$\mathbb{S}^{n+1}$ is not an n-type for any $n \geq -1$.*

C 8.6.14. *$\pi_3(\mathbb{S}^2) = \mathbb{Z}$.*

8.7 The van Kampen theorem

8.7.1 Naive van Kampen

NB 8.7.1. One might expect to see in the definition of code some additional generating equations for the set-quotient, such as

$$\begin{aligned} (\ldots, p_{k-1} \cdot fw, x_k', q_k, \ldots) &= (\ldots, p_{k-1}, x_k, gw \cdot q_k, \ldots) && (\text{for } w : \Pi_1 A(x_k, x_k')) \\ (\ldots, q_k \cdot gw, y_k', p_k, \ldots) &= (\ldots, q_k, y_k, fw \cdot p_k, \ldots). && (\text{for } w : \Pi_1 A(y_k, y_k')) \end{aligned}$$

However, these are not necessary! In fact, they follow automatically by path induction on w. This is the main difference between the "naive" van Kampen theorem and the more refined one we will consider in the next subsection.

T 8.7.2 (Naive van Kampen theorem)**.** *For all $u, v : P$ there is an equivalence*

$$\Pi_1 P(u,v) \simeq \mathsf{code}(u,v).$$

E 8.7.3. Let $A :\equiv \mathbf{2}$, $B :\equiv \mathbf{1}$, and $C :\equiv \mathbf{1}$. Then $P \simeq S^1$. Inspecting the definition of, say, $\mathsf{code}(i(\star), i(\star))$, we see that the paths all may as well be trivial, so the only information is in the sequence of elements $x_1, y_1, \ldots, x_n, y_n : \mathbf{2}$. Moreover, if we have $x_k = y_k$ or $y_k = x_{k+1}$ for any k, then the set-quotient relations allow us to excise both of those elements. Thus, every such sequence is equal to a canonical *reduced* one in which no two adjacent elements are equal. Clearly such a reduced sequence is uniquely determined by its length (a natural number n) together with, if $n > 1$, the information of whether x_1 is $0_\mathbf{2}$ or $1_\mathbf{2}$, since that determines the rest of the sequence uniquely. And these data can, of course, be identified with an integer, where n is the absolute value and x_1 encodes the sign. Thus we recover $\pi_1(S^1) \cong \mathbb{Z}$.

E 8.7.4. More generally, let $B :\equiv \mathbf{1}$ and $C :\equiv \mathbf{1}$ but A be arbitrary, so that P is the suspension of A. Then once again the paths p_k and q_k are trivial, so that the only information in a path code is a sequence of elements $x_1, y_1, \ldots, x_n, y_n : A$. The first two generating equalities say that adjacent equal elements can be canceled, so it makes sense to think of this sequence as a word of the form

$$x_1 y_1^{-1} x_2 y_2^{-1} \cdots x_n y_n^{-1}$$

in a group. Indeed, it looks similar to the free group on A (or equivalently on $\|A\|_0$; see **??**), but we are considering only words that start with a non-inverted element, alternate between inverted and non-inverted elements, and end with an inverted one. This effectively reduces the size of the generating set by one. For instance, if A has a point $a : A$, then we can identify $\pi_1(\Sigma A)$ with the group presented by $\|A\|_0$ as generators with the relation $|a|_0 = e$; see **????** for details.

E 8.7.5. Let $A :\equiv \mathbf{1}$ and B and C be arbitrary, so that f and g simply equip B and C with basepoints b and c, say. Then P is the *wedge* $B \vee C$ of B and C (the coproduct in the category of based spaces). In this case, it is the elements x_k and y_k which are trivial, so that the only information is a sequence of loops $(p_0, q_1, p_1, \ldots, p_n)$ with $p_k : \pi_1(B, b)$ and $q_k : \pi_1(C, c)$. Such sequences, modulo the equivalence relation we have imposed, are easily identified with the explicit description of the *free product* of the groups $\pi_1(B, b)$ and $\pi_1(C, c)$, as constructed in **??**. Thus, we have $\pi_1(B \vee C) \cong \pi_1(B) * \pi_1(C)$.

8.7.2 The van Kampen theorem with a set of basepoints

L 8.7.6. *$\bar{k}$ is 0-connected.*

NB 8.7.7. T can be regarded as the (homotopy) coequalizer of the "kernel pair" of k. If S and A were sets, then the (-1)-connectivity of k would imply that A is the 0-truncation of this coequalizer (see **??**). For general types, higher topos theory suggests that (-1)-connectivity of k will imply instead that A is the colimit (a.k.a. "geometric realization") of the "simplicial kernel" of k. The type T is the colimit of the "1-skeleton" of this simplicial kernel, so it makes sense that it improves the connectivity of k by 1. More generally, we might expect the colimit of the n-skeleton to improve connectivity by n.

T 8.7.8 (van Kampen with a set of basepoints)**.** *For all $u, v : P$ there is an equivalence*

$$\Pi_1 P(u,v) \simeq \mathsf{code}(u,v).$$

with code defined as in this section.

E 8.7.9. Suppose $S :\equiv \mathbf{1}$, so that A has a basepoint $a :\equiv k(\star)$ and is connected. Then code for loops in the pushout can be identified with alternating sequences of loops in $\pi_1(B, f(a))$ and $\pi_1(C, g(a))$, modulo an equivalence relation which allows us to slide elements of $\pi_1(A, a)$ between them (after applying f and g respectively). Thus, $\pi_1(P)$ can be identified with the *amalgamated free product* $\pi_1(B) *_{\pi_1(A)} \pi_1(C)$ (the pushout in the category of groups), as constructed in **??**. This (in the case when B and C are open subspaces of P and A their intersection) is probably the most classical version of the van Kampen theorem.

E 8.7.10. As a special case of **??**, suppose additionally that $C :\equiv \mathbf{1}$, so that P is the cofiber B/A. Then every loop in C is equal to reflexivity, so the relations on path codes allow us to collapse all sequences to a single loop in B. The additional relations require that multiplying on the left, right, or in the middle by an element in the image of $\pi_1(A)$ is the identity. We can thus identify $\pi_1(B/A)$ with the quotient of the group $\pi_1(B)$ by the normal subgroup generated by the image of $\pi_1(A)$.

E 8.7.11. As a further special case of **??**, let $B :\equiv S^1 \vee S^1$, let $A :\equiv S^1$, and let $f : A \to B$ pick out the composite loop $p \cdot q \cdot p^{-1} \cdot q^{-1}$, where p and q are the generating loops in the two copies of S^1 comprising B. Then P is a presentation of the torus T^2. Indeed, it is not hard to identify P with the presentation of T^2 as described in **??**, using the cone on a particular loop. Thus, $\pi_1(T^2)$ is the quotient of the free group on two generators (i.e., $\pi_1(B)$) by the relation $p \cdot q \cdot p^{-1} \cdot q^{-1} = 1$. This clearly yields the free *abelian* group on two generators, which is $\mathbb{Z} \times \mathbb{Z}$.

E 8.7.12. More generally, any CW complex can be obtained by repeatedly "coning off" spheres, as described in **??**. That is, we start with a set X_0 of points ("0-cells"), which is the "0-skeleton" of the CW complex. We take the pushout

$$\begin{tikzcd}
S_1 \times \mathbb{S}^0 \arrow[r, "f_1"] \arrow[d] & X_0 \arrow[d] \\
\mathbf{1} \arrow[r] & X_1
\end{tikzcd}$$

for some set S_1 of 1-cells and some family f_1 of "attaching maps", obtaining the "1-skeleton" X_1. Then we take the pushout

$$\begin{tikzcd}
S_2 \times \mathbb{S}^1 \arrow[r, "f_2"] \arrow[d] & X_1 \arrow[d] \\
\mathbf{1} \arrow[r] & X_2
\end{tikzcd}$$

for some set S_2 of 2-cells and some family f_2 of attaching maps, obtaining the 2-skeleton X_2, and so on. The fundamental group of each pushout can be calculated from the van Kampen theorem: we obtain the group presented by generators derived from the 1-skeleton, and relations derived from S_2 and f_2. The pushouts after this stage do not alter the fundamental group, since $\pi_1(\mathbb{S}^n)$ is trivial for $n > 1$ (see **??**).

E 8.7.13. In particular, suppose given any presentation of a (set-)group $G = \langle X \mid R \rangle$, with X a set of generators and R a set of words in these generators. Let $B :\equiv \bigvee_X S^1$ and $A :\equiv \bigvee_R S^1$, with $f : A \to B$ sending each copy of S^1 to the corresponding word in the generating loops of B. It follows that $\pi_1(P) \cong G$; thus we have constructed a connected type whose fundamental group is G. Since any group has a presentation, any group is the fundamental group of some type. If we 1-truncate such a type, we obtain a type whose only nontrivial homotopy group is G; this is called an **Eilenberg–Mac Lane space** $K(G, 1)$.

8.8 Whitehead's theorem and Whitehead's principle

T 8.8.1. *Suppose $f : A \to B$ is a function such that*

(i). $\|f\|_0 : \|A\|_0 \to \|B\|_0$ *is surjective, and*
(ii). *for any $x, y : A$, the function $\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y))$ is an equivalence.*

Then f is an equivalence.

C 8.8.2. *Suppose $f : A \to B$ is a function such that*

(i). $\|f\|_0 : \|A\|_0 \to \|B\|_0$ *is a bijection, and*
(ii). *for any $x : A$, the function $\Omega f : \Omega(A, x) \to \Omega(B, f(x))$ is an equivalence.*

Then f is an equivalence.

T 8.8.3. *Suppose A and B are n-types and $f : A \to B$ is such that*

(i). $\|f\|_0 : \|A\|_0 \to \|B\|_0$ *is a bijection, and*
(ii). $\pi_k(f) : \pi_k(A, x) \to \pi_k(B, f(x))$ *is a bijection for all $k \geq 1$ and all $x : A$.*

Then f is an equivalence.

C 8.8.4. *If A is a 0-connected n-type and $\pi_k(A, a) = 0$ for all k and $a : A$, then A is contractible.*

C 8.8.5. *For $n \geq 0$, a map $f : A \to B$ is n-connected if and only if the following all hold:*

(i). $\|f\|_0 : \|A\|_0 \to \|B\|_0$ *is an isomorphism.*
(ii). *For any $a : A$ and $k \leq n$, the map $\pi_k(f) : \pi_k(A, a) \to \pi_k(B, f(a))$ is an isomorphism.*
(iii). *For any $a : A$, the map $\pi_{n+1}(f) : \pi_{n+1}(A, a) \to \pi_{n+1}(B, f(a))$ is surjective.*

E 8.8.6. Suppose we have $B : \mathbb{N} \to \mathcal{U}$ such that for each n, the type $B(n)$ contains an n-loop which is not equal to n-fold reflexivity, say $p_n : \Omega^n(B(n), b_n)$ with $p_n \neq \mathsf{refl}^n_{b_n}$. (For instance, we could define $B(n) :\equiv \mathbb{S}^n$, with p_n the image of $1 : \mathbb{Z}$ under the isomorphism $\pi_n(\mathbb{S}^n) \cong \mathbb{Z}$.) Consider $C :\equiv \prod_{(n:\mathbb{N})} B(n)$, with the point $c : C$ defined by $c(n) :\equiv b_n$. Since loop spaces commute with products, for any m we have

$$\Omega^m(C, c) \ \simeq \ \prod_{n:\mathbb{N}} \Omega^m(B(n), b_n).$$

Under this equivalence, refl^m_c corresponds to the function $(n \mapsto \mathsf{refl}^m_{b_n})$. Now define q_m in the right-hand type by

$$q_m(n) :\equiv \begin{cases} p_n & m = n \\ \mathsf{refl}^m_{b_n} & m \neq n. \end{cases}$$

If we had $q_m = (n \mapsto \mathsf{refl}^m_{b_n})$, then we would have $p_n = \mathsf{refl}^n_{b_n}$, which is not the case. Thus, $q_m \neq (n \mapsto \mathsf{refl}^m_{b_n})$, and so there is a point of $\Omega^m(C, c)$ which is unequal to refl^m_c. Hence C is not an m-type, for any $m : \mathbb{N}$.

8.9 A general statement of the encode-decode method

L 8.9.1 (Encode-decode for Loop Spaces). *Given a pointed type (A, a_0) and a fibration $\mathsf{code} : A \to \mathcal{U}$, if*

(i). $c_0 : \mathsf{code}(a_0)$,
(ii). $\mathsf{decode} : \prod_{(x:A)} \mathsf{code}(x) \to (a_0 = x)$,
(iii). *for all $c : \mathsf{code}(a_0)$, $\mathsf{transport}^{\mathsf{code}}(\mathsf{decode}(c), c_0) = c$, and*
(iv). $\mathsf{decode}(c_0) = \mathsf{refl}$,

then $(a_0 = a_0)$ is equivalent to $\mathsf{code}(a_0)$.

L 8.9.2 (Encode-decode for Truncations of Loop Spaces). *Assume a pointed type (A, a_0) and a fibration $\mathsf{code} : A \to \mathcal{U}$, where for every x, $\mathsf{code}(x)$ is a k-type. Define*

$$\mathsf{encode} : \prod_{x:A} \|a_0 = x\|_k \to \mathsf{code}(x)$$

by truncation recursion (using the fact that $\mathsf{code}(x)$ is a k-type), mapping $\alpha : a_0 = x$ to $\mathsf{transport}^{\mathsf{code}}(\alpha, c_0)$. Suppose:

(i). $c_0 : \mathsf{code}(a_0)$,
(ii). $\mathsf{decode} : \prod_{(x:A)} \mathsf{code}(x) \to \|a_0 = x\|_k$,
(iii). $\mathsf{encode}_{a_0}(\mathsf{decode}_{a_0}(c)) = c$ *for all $c : \mathsf{code}(a_0)$, and*
(iv). $\mathsf{decode}(c_0) = |\mathsf{refl}|$.

Then $\|a_0 = a_0\|_k$ is equivalent to $\mathsf{code}(a_0)$.

8.10 Additional Results

T 8.10.1. *There exists a k such that for all $n \geq 3$, $\pi_{n+1}(\mathbb{S}^n) = \mathbb{Z}_k$.*

T 8.10.2 (Blakers–Massey theorem)**.** *Suppose we are given maps $f : C \to X$, and $g : C \to Y$. Taking first the pushout $X \sqcup^C Y$ of f and g and then the pullback of its inclusions $\mathsf{inl} : X \to X \sqcup^C Y \leftarrow Y : \mathsf{inr}$, we have an induced map $C \to X \times_{(X \sqcup^C Y)} Y$.*
If f is i-connected and g is j-connected, then this induced map is $(i+j)$-connected. In other words, for any points $x : X$, $y : Y$, the corresponding fiber $C_{x,y}$ of $(f, g) : C \to X \times Y$ gives an approximation to the path space $\mathsf{inl}(x) =_{X \sqcup^C Y} \mathsf{inr}(y)$ in the pushout.

T 8.10.3 (Eilenberg–Mac Lane Spaces)**.** *For any abelian group G and positive integer n, there is an n-type $K(G, n)$ such that $\pi_n(K(G, n)) = G$, and $\pi_k(K(G, n)) = 0$ for $k \neq n$.*

T 8.10.4 (Covering spaces)**.** *For a connected space A, there is an equivalence between covering spaces over A and sets with an action of $\pi_1(A)$.*

Notes

Exercises

ex Prove that homotopy groups respect products: $\pi_n(A \times B) \simeq \pi_n(A) \times \pi_n(B)$.

Exer. 8.81. Prove that if A is a set with decidable equality (see **??**), then its suspension ΣA is a 1-type. (It is an open question whether this is provable without the assumption of decidable equality.)

Exer. 8.82. Define $\mathbb{S}^\infty$ to be the colimit of the sequence $\mathbb{S}^0 \to \mathbb{S}^1 \to \mathbb{S}^2 \to \cdots$. Prove that $\mathbb{S}^\infty$ is contractible.

Exer. 8.83. Define $\mathbb{S}^\infty$ to be the higher inductive type generated by

- Two points $\mathsf{N} : \mathbb{S}^\infty$ and $\mathsf{S} : \mathbb{S}^\infty$, and
- For each $x : \mathbb{S}^\infty$, a path $\mathsf{merid}(x) : \mathsf{N} = \mathsf{S}$.

In other words, $\mathbb{S}^\infty$ is its own suspension. Prove that $\mathbb{S}^\infty$ is contractible.

Exer. 8.84. Suppose $f : X \to Y$ is a function and Y is connected. Show that for any $y_1, y_2 : Y$ we have $\left\| \mathsf{fib}_f(y_1) \simeq \mathsf{fib}_f(y_2) \right\|$.

Exer. 8.85. For any pointed type A, let $i_A : \Omega A \to \Omega A$ denote inversion of loops, $i_A :\equiv \lambda p.\, p^{-1}$. Show that $i_{\Omega A} : \Omega^2 A \to \Omega^2 A$ is equal to $\Omega(i_A)$.

Exer. 8.86. Define a **pointed equivalence** to be a pointed map whose underlying function is an equivalence.

(i). Show that the type of pointed equivalences between pointed types (X, x_0) and (Y, y_0) is equivalent to $(X, x_0) =_{\mathcal{U}_\bullet} (Y, y_0)$.
(ii). Reformulate the notion of pointed equivalence in terms of a pointed quasi-inverse and pointed homotopies, in one of the coherent styles from **??**.

Exer. 8.87. Following the example of the Hopf fibration in **??**, define the **junior Hopf fibration** as a fibration (that is, a type family) over $\mathbb{S}^1$ whose fiber over the basepoint is $\mathbb{S}^0$ and whose total space is $\mathbb{S}^1$. This is also called the "twisted double cover" of the circle $\mathbb{S}^1$.

Exer. 8.88. Again following the example of the Hopf fibration in **??**, define an analogous fibration over $\mathbb{S}^4$ whose fiber over the basepoint is $\mathbb{S}^3$ and whose total space is $\mathbb{S}^7$. This is an open problem in homotopy type theory (such a fibration is known to exist in classical homotopy theory).

Exer. 8.89. Continuing from **??**, prove that if A has a point $a : A$, then we can identify $\pi_1(\Sigma A)$ with the group presented by $\|A\|_0$ as generators with the relation $|a|_0 = e$. Then show that if we assume excluded middle, this is also the free group on $\|A\|_0 \setminus \{|a|_0\}$.

Exer. 8.90. Again continuing from **??**, but this time without assuming A to be pointed, show that we can identify $\pi_1(\Sigma A)$ with the group presented by generators $\|A\|_0 \times \|A\|_0$ and relations

$$(a, b) = (b, a)^{-1}, \qquad (a, c) = (a, b) \cdot (b, c), \qquad \text{and} \qquad (a, a) = e.$$

9.1 Categories and precategories

D 9.1.1. A **precategory** A consists of the following.

(i). A type A_0, whose elements are called **objects**. We write $a : A$ for $a : A_0$.
(ii). For each $a, b : A$, a set $\hom_A(a, b)$, whose elements are called **arrows** or **morphisms**.
(iii). For each $a : A$, a morphism $1_a : \hom_A(a, a)$, called the **identity morphism**.
(iv). For each $a, b, c : A$, a function

$$\hom_A(b, c) \to \hom_A(a, b) \to \hom_A(a, c)$$

called **composition**, and denoted infix by $g \mapsto f \mapsto g \circ f$, or sometimes simply by gf.
(v). For each $a, b : A$ and $f : \hom_A(a, b)$, we have $f = 1_b \circ f$ and $f = f \circ 1_a$.
(vi). For each $a, b, c, d : A$ and

$$f : \hom_A(a, b), \qquad g : \hom_A(b, c), \qquad h : \hom_A(c, d),$$

we have $h \circ (g \circ f) = (h \circ g) \circ f$.

D 9.1.2. A morphism $f : \hom_A(a, b)$ is an **isomorphism** if there is a morphism $g : \hom_A(b, a)$ such that $g \circ f = 1_a$ and $f \circ g = 1_b$. We write $a \cong b$ for the type of such isomorphisms.

L 9.1.3. *For any $f : \hom_A(a, b)$, the type "f is an isomorphism" is a mere proposition. Therefore, for any $a, b : A$ the type $a \cong b$ is a set.*

L 9.1.4 (idtoiso). *If A is a precategory and $a, b : A$, then*

$$(a = b) \to (a \cong b).$$

E 9.1.5. There is a precategory $\mathcal{S}et$, whose type of objects is Set, and with $\hom_{\mathcal{S}et}(A, B) :\equiv (A \to B)$. The identity morphisms are identity functions and the composition is function composition. For this precategory, **??** is equal to (the restriction to sets of) the map $\mathsf{idtoeqv}$ from **??**.
Of course, to be more precise we should call this category $\mathcal{S}et_{\mathcal{U}}$, since its objects are only the *small sets* relative to a universe $\mathcal{U}$.

D 9.1.6. A **category** is a precategory such that for all $a, b : A$, the function $\mathsf{idtoiso}_{a,b}$ from **??** is an equivalence.

E 9.1.7. The univalence axiom implies immediately that $\mathcal{S}et$ is a category. One can also show, using univalence, that any precategory of set-level structures such as groups, rings, topological spaces, etc. is a category; see **??**.

L 9.1.8. *In a category, the type of objects is a 1-type.*

L 9.1.9. *For $p : a = a'$ and $q : b = b'$ and $f : \hom_A(a, b)$, we have*

$$(p, q)_*(f) = \mathsf{idtoiso}(q) \circ f \circ \mathsf{idtoiso}(p)^{-1}. \tag{9.1.10}$$

E 9.1.11. A precategory in which each set $\hom_A(a, b)$ is a mere proposition is equivalently a type A_0 equipped with a mere relation "$\le$" that is reflexive ($a \le a$) and transitive (if $a \le b$ and $b \le c$, then $a \le c$). We call this a **preorder**.
In a preorder, a witness $f : a \le b$ is an isomorphism just when there exists some witness $g : b \le a$. Thus, $a \cong b$ is the mere proposition that $a \le b$ and $b \le a$. Therefore, a preorder A is a category just when (1) each type $a = b$ is a mere proposition, and (2) for any $a, b : A_0$ there exists a function $(a \cong b) \to (a = b)$. In other words, A_0 must be a set, and $\le$ must be antisymmetric (if $a \le b$ and $b \le a$, then $a = b$). We call this a **(partial) order** or a **poset**.

E 9.1.12. If A is a category, then A_0 is a set if and only if for any $a, b : A_0$, the type $a \cong b$ is a mere proposition. This is equivalent to saying that every isomorphism in A is an identity; thus it is rather stronger than the classical notion of "skeletal" category. Categories of this sort are sometimes called **gaunt** [**?**]. There is not really any notion of "skeletality" for our categories, unless one considers **??** itself to be such.

E 9.1.13. For any 1-type X, there is a category with X as its type of objects and with $\hom(x, y) :\equiv (x = y)$. If X is a set, we call this the **discrete** category on X. In general, we call it a **groupoid** (see **??**).

E 9.1.14. For *any* type X, there is a precategory with X as its type of objects and with $\hom(x, y) :\equiv \|x = y\|_0$. The composition operation

$$\|y = z\|_0 \to \|x = y\|_0 \to \|x = z\|_0$$

is defined by induction on truncation from concatenation $(y = z) \to (x = y) \to (x = z)$. We call this the **fundamental pregroupoid** of X. (In fact, we have met it already in **??**; see also **??**.)

E 9.1.15. There is a precategory whose type of objects is $\mathcal{U}$ and with $\hom(X, Y) :\equiv \|X \to Y\|_0$, and composition defined by induction on truncation from ordinary composition $(Y \to Z) \to (X \to Y) \to (X \to Z)$. We call this the **homotopy precategory of types**.

E 9.1.16. Let $\mathcal{R}el$ be the following precategory:

- Its objects are sets.
- $\hom_{\mathcal{R}el}(X, Y) = X \to Y \to \mathsf{Prop}$.
- For a set X, we have $1_X(x, x') :\equiv (x = x')$.
- For $R : \hom_{\mathcal{R}el}(X, Y)$ and $S : \hom_{\mathcal{R}el}(Y, Z)$, their composite is defined by
$$(S \circ R)(x, z) :\equiv \left\| \sum_{y:Y} R(x, y) \times S(y, z) \right\|.$$

Suppose $R : \hom_{\mathcal{R}el}(X, Y)$ is an isomorphism, with inverse S. We observe the following.

(i). If $R(x, y)$ and $S(y', x)$, then $(R \circ S)(y', y)$, and hence $y' = y$. Similarly, if $R(x, y)$ and $S(y, x')$, then $x = x'$.
(ii). For any x, we have $x = x$, hence $(S \circ R)(x, x)$. Thus, there merely exists a $y : Y$ such that $R(x, y)$ and $S(y, x)$.
(iii). Suppose $R(x, y)$. By (, there merely exists a y' with $R(x, y')$ and $S(y', x)$. But then by (, merely $y' = y$, and hence $y' = y$ since Y is a set. Therefore, by transporting $S(y', x)$ along this equality, we have $S(y, x)$. In conclusion, $R(x, y) \to S(y, x)$. Similarly, $S(y, x) \to R(x, y)$.
(iv). If $R(x, y)$ and $R(x, y')$, then by (, $S(y', x)$, so that by (, $y = y'$. Thus, for any x there is at most one y such that $R(x, y)$. And by (, there merely exists such a y, hence there exists such a y.

In conclusion, if $R : \hom_{\mathcal{R}el}(X, Y)$ is an isomorphism, then for each $x : X$ there is exactly one $y : Y$ such that $R(x, y)$, and dually. Thus, there is a function $f : X \to Y$ sending each x to this y, which is an equivalence; hence $X = Y$. With a little more work, we conclude that $\mathcal{R}el$ is a category.

9.2 Functors and transformations

D 9.2.1. Let A and B be precategories. A **functor** $F : A \to B$ consists of

(i). A function $F_0 : A_0 \to B_0$, generally also denoted F.
(ii). For each $a, b : A$, a function $F_{a,b} : \hom_A(a, b) \to \hom_B(Fa, Fb)$, generally also denoted F.
(iii). For each $a : A$, we have $F(1_a) = 1_{Fa}$.
(iv). For each $a, b, c : A$ and $f : \hom_A(a, b)$ and $g : \hom_A(b, c)$, we have
$$F(g \circ f) = Fg \circ Ff.$$

D 9.2.2. For functors $F, G : A \to B$, a **natural transformation** $\gamma : F \to G$ consists of

(i). For each $a : A$, a morphism $\gamma_a : \hom_B(Fa, Ga)$ (the "components").
(ii). For each $a, b : A$ and $f : \hom_A(a, b)$, we have $Gf \circ \gamma_a = \gamma_b \circ Ff$ (the "naturality axiom").

D 9.2.3. For precategories A, B, there is a precategory B^A, called the **functor precategory**, defined by

- $(B^A)_0$ is the type of functors from A to B.
- $\hom_{B^A}(F, G)$ is the type of natural transformations from F to G.

L 9.2.4. *A natural transformation $\gamma : F \to G$ is an isomorphism in B^A if and only if each γ_a is an isomorphism in B.*

T 9.2.5. *If A is a precategory and B is a category, then B^A is a category.*

D 9.2.6. For functors $F : A \to B$ and $G : B \to C$, their composite $G \circ F : A \to C$ is given by

- The composite $(G_0 \circ F_0) : A_0 \to C_0$
- For each $a, b : A$, the composite
$$(G_{Fa,Fb} \circ F_{a,b}) : \hom_A(a, b) \to \hom_C(GFa, GFb).$$

It is easy to check the axioms.

D 9.2.7. For functors $F : A \to B$ and $G, H : B \to C$ and a natural transformation $\gamma : G \to H$, the composite $(\gamma F) : GF \to HF$ is given by

- For each $a : A$, the component γ_{Fa}.

Naturality is easy to check. Similarly, for γ as above and $K : C \to D$, the composite $(K\gamma) : KG \to KH$ is given by

- For each $b : B$, the component $K(\gamma_b)$.

L 9.2.8. *For functors $F, G : A \to B$ and $H, K : B \to C$ and natural transformations $\gamma : F \to G$ and $\delta : H \to K$, we have*
$$(\delta G)(H\gamma) = (K\gamma)(\delta F).$$

L 9.2.9. *Composition of functors is associative:* $H(GF) = (HG)F$.

L 9.2.10. **??** *is coherent, i.e. the following pentagon of equalities commutes:*

$$\begin{tikzcd} & K(H(GF)) \arrow[dl, equal] \arrow[dr, equal] & \\ (KH)(GF) \arrow[d, equal] & & K((HG)F) \arrow[d, equal] \\ ((KH)G)F \arrow[rr, equal] & & (K(HG))F \end{tikzcd}$$

L 9.2.11. *For a functor* $F : A \to B$*, we have equalities* $(1_B \circ F) = F$ *and* $(F \circ 1_A) = F$*, such that given also* $G : B \to C$*, the following triangle of equalities commutes.*

$$\begin{tikzcd} G \circ (1_B \circ F) \arrow[rr, equal] \arrow[dr, equal] & & (G \circ 1_B) \circ F \arrow[dl, equal] \\ & G \circ F. & \end{tikzcd}$$

9.3 Adjunctions

D 9.3.1. A functor $F : A \to B$ is a **left adjoint** if there exists

- A functor $G : B \to A$.
- A natural transformation $\eta : 1_A \to GF$ (the **unit**).
- A natural transformation $\epsilon : FG \to 1_B$ (the **counit**).
- $(\epsilon F)(F\eta) = 1_F$.
- $(G\epsilon)(\eta G) = 1_G$.

L 9.3.2. *If* A *is a category (but* B *may be only a precategory), then the type "*F *is a left adjoint" is a mere proposition.*

9.4 Equivalences

D 9.4.1. A functor $F : A \to B$ is an **equivalence of (pre)categories** if it is a left adjoint for which η and ϵ are isomorphisms. We write $A \simeq B$ for the type of equivalences of categories from A to B.

L 9.4.2. *If for* $F : A \to B$ *there exists* $G : B \to A$ *and isomorphisms* $GF \cong 1_A$ *and* $FG \cong 1_B$*, then* F *is an equivalence of precategories.*

D 9.4.3. We say a functor $F : A \to B$ is **faithful** if for all $a, b : A$, the function

$$F_{a,b} : \hom_A(a, b) \to \hom_B(Fa, Fb)$$

is injective, and **full** if for all $a, b : A$ this function is surjective. If it is both (hence each $F_{a,b}$ is an equivalence) we say F is **fully faithful**.

D 9.4.4. We say a functor $F : A \to B$ is **split essentially surjective** if for all $b : B$ there exists an $a : A$ such that $Fa \cong b$.

L 9.4.5. *For any precategories* A *and* B *and functor* $F : A \to B$*, the following types are equivalent.*

(i). F *is an equivalence of precategories.*
(ii). F *is fully faithful and split essentially surjective.*

D 9.4.6. A functor $F : A \to B$ is **essentially surjective** if for all $b : B$, there *merely* exists an $a : A$ such that $Fa \cong b$. We say F is a **weak equivalence** if it is fully faithful and essentially surjective.

L 9.4.7. *If* $F : A \to B$ *is fully faithful and* A *is a category, then for any* $b : B$ *the type* $\sum_{(a:A)} (Fa \cong b)$ *is a mere proposition. Hence a functor between categories is an equivalence if and only if it is a weak equivalence.*

D 9.4.8. A functor $F : A \to B$ is an **isomorphism of (pre)categories** if F is fully faithful and $F_0 : A_0 \to B_0$ is an equivalence of types.

L 9.4.9. *For precategories A and B and $F : A \to B$, the following are equivalent.*

(i). *F is an isomorphism of precategories.*

(ii). *There exist $G : B \to A$ and $\eta : 1_A = GF$ and $\epsilon : FG = 1_B$ such that*

$$\mathsf{ap}_{(\lambda H.\, FH)}(\eta) = \mathsf{ap}_{(\lambda K.\, KF)}(\epsilon^{-1}). \tag{9.4.10}$$

(iii). *There merely exist $G : B \to A$ and $\eta : 1_A = GF$ and $\epsilon : FG = 1_B$.*

Proof. First note that since hom-sets are sets, equalities between equalities of functors are uniquely determined by their object-parts. Thus, by function extensionality, (9) is equivalent to

$$(F_0)(\eta_0)_a = (\epsilon_0)^{-1}{}_{F_0 a}. \tag{9.4.11}$$

for all $a : A_0$. Note that this is precisely the triangle identity for G_0, η_0, and ϵ_0 to be a proof that F_0 is a half adjoint equivalence of types.
Now suppose (. Let $G_0 : B_0 \to A_0$ be the inverse of F_0, with $\eta_0 : \mathsf{id}_{A_0} = G_0F_0$ and $\epsilon_0 : F_0G_0 = \mathsf{id}_{B_0}$ satisfying the triangle identity, which is precisely (9). Now define $G_{b,b'} : \hom_B(b,b') \to \hom_A(G_0b, G_0b')$ by

$$G_{b,b'}(g) :\equiv (F_{G_0b,G_0b'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\Big)$$

(using the assumption that F is fully faithful). Since idtoiso takes inverses to inverses and concatenation to composition, and F is a functor, it follows that G is a functor.
By definition, we have $(GF)_0 \equiv G_0F_0$, which is equal to id_{A_0} by η_0. To obtain $1_A = GF$, we need to show that when transported along η_0, the identity function of $\hom_A(a,a')$ becomes equal to the composite $G_{Fa,Fa'} \circ F_{a,a'}$. In other words, for any $f : \hom_A(a,a')$ we must have

$$\mathsf{idtoiso}((\eta_0)_{a'}) \circ f \circ \mathsf{idtoiso}((\eta_0)^{-1}{}_a) = (F_{GFa,GFa'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{Fa'}) \circ F_{a,a'}(f) \circ \mathsf{idtoiso}((\epsilon_0)_{Fa})\Big).$$

But this is equivalent to

$$(F_{GFa,GFa'})\Big(\mathsf{idtoiso}((\eta_0)_{a'}) \circ f \circ \mathsf{idtoiso}((\eta_0)^{-1}{}_a)\Big) = \mathsf{idtoiso}((\epsilon_0)^{-1}{}_{Fa'}) \circ F_{a,a'}(f) \circ \mathsf{idtoiso}((\epsilon_0)_{Fa}).$$

which follows from functoriality of F, the fact that F preserves idtoiso, and (9). Thus we have $\eta : 1_A = GF$.
On the other side, we have $(FG)_0 \equiv F_0G_0$, which is equal to id_{B_0} by ϵ_0. To obtain $FG = 1_B$, we need to show that when transported along ϵ_0, the identity function of $\hom_B(b,b')$ becomes equal to the composite $F_{Gb,Gb'} \circ G_{b,b'}$. That is, for any $g : \hom_B(b,b')$ we must have

$$F_{Gb,Gb'}\Big((F_{Gb,Gb'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\Big)\Big) = \mathsf{idtoiso}((\epsilon_0{}^{-1})_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b).$$

But this is just the fact that $(F_{Gb,Gb'})^{-1}$ is the inverse of $F_{Gb,Gb'}$. And we have remarked that (9) is equivalent to (9), so (holds.
Conversely, suppose given (; then the object-parts of G, η, and ϵ together with (9) show that F_0 is an equivalence of types. And for $a, a' : A_0$, we define $\overline{G}_{a,a'} : \hom_B(Fa, Fa') \to \hom_A(a,a')$ by

$$\overline{G}_{a,a'}(g) :\equiv \mathsf{idtoiso}(\eta^{-1})_{a'} \circ G(g) \circ \mathsf{idtoiso}(\eta)_a. \tag{9.4.12}$$

By naturality of $\mathsf{idtoiso}(\eta)$, for any $f : \hom_A(a,a')$ we have

$$\begin{aligned}\overline{G}_{a,a'}(F_{a,a'}(f)) &= \mathsf{idtoiso}(\eta^{-1})_{a'} \circ G(F(f)) \circ \mathsf{idtoiso}(\eta)_a \\ &= \mathsf{idtoiso}(\eta^{-1})_{a'} \circ \mathsf{idtoiso}(\eta)_{a'} \circ f \\ &= f.\end{aligned}$$

On the other hand, for $g : \hom_B(Fa, Fa')$ we have

$$\begin{aligned}F_{a,a'}(\overline{G}_{a,a'}(g)) &= F(\mathsf{idtoiso}(\eta^{-1})_{a'}) \circ F(G(g)) \circ F(\mathsf{idtoiso}(\eta)_a) \\ &= \mathsf{idtoiso}(\epsilon)_{Fa'} \circ F(G(g)) \circ \mathsf{idtoiso}(\epsilon^{-1})_{Fa} \\ &= \mathsf{idtoiso}(\epsilon)_{Fa'} \circ \mathsf{idtoiso}(\epsilon^{-1})_{Fa'} \circ g \\ &= g.\end{aligned}$$

(There are lemmas needed here regarding the compatibility of idtoiso and whiskering, which we leave it to the reader to state and prove.) Thus, $F_{a,a'}$ is an equivalence, so F is fully faithful; i.e. (holds.
Now the composite ($\to$($\to$(is equal to the identity since (is a mere proposition. On the other side, tracing through the above constructions we see that the composite ($\to$($\to$(essentially preserves the object-parts G_0, η_0, ϵ_0, and the object-part of (9). And in the latter three cases, the object-part is all there is, since hom-sets are sets.
Thus, it suffices to show that we recover the action of G on hom-sets. In other words, we must show that if $g : \hom_B(b, b')$, then

$$G_{b,b'}(g) = \overline{G}_{G_0 b, G_0 b'}\left(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\right)$$

where $\overline{G}$ is defined by (9). However, this follows from functoriality of G and the *other* triangle identity, which we have seen in **??** is equivalent to (9).
Now since (is a mere proposition, so is (, so it suffices to show they are logically equivalent to (. Of course, ($\to$(, so let us assume (. Since (is a mere proposition, we may assume given G, η, and ϵ. Then G_0 along with η and ϵ imply that F_0 is an equivalence. Moreover, we also have natural isomorphisms $\mathsf{idtoiso}(\eta) : 1_A \cong GF$ and $\mathsf{idtoiso}(\epsilon) : FG \cong 1_B$, so by **??**, F is an equivalence of precategories, and in particular fully faithful. □

E 9.4.13. Let X be a type and $x_0 : X$ an element, and let X_{ch} denote the *chaotic* or *indiscrete* precategory on X. By definition, we have $(X_{\mathrm{ch}})_0 :\equiv X$, and $\hom_{X_{\mathrm{ch}}}(x, x') :\equiv \mathbf{1}$ for all x, x'. Then the unique functor $X_{\mathrm{ch}} \to \mathbf{1}$ is an equivalence of precategories, but not an isomorphism unless X is contractible.
This example also shows that a precategory can be equivalent to a category without itself being a category. Of course, if a precategory is *isomorphic* to a category, then it must itself be a category.

L 9.4.14. *For categories A and B, a functor $F : A \to B$ is an equivalence of categories if and only if it is an isomorphism of categories.*

L 9.4.15. *If A and B are precategories, then the function*

$$(A = B) \to (A \cong B)$$

(defined by induction from the identity functor) is an equivalence of types.

T 9.4.16. *If A and B are categories, then the function*

$$(A = B) \to (A \simeq B)$$

(defined by induction from the identity functor) is an equivalence of types.

9.5 The Yoneda lemma

D 9.5.1. For a precategory A, its **opposite** A^{op} is a precategory with the same type of objects, with $\hom_{A^{\mathrm{op}}}(a, b) :\equiv \hom_A(b, a)$, and with identities and composition inherited from A.

D 9.5.2. For precategories A and B, their **product** $A \times B$ is a precategory with $(A \times B)_0 :\equiv A_0 \times B_0$ and

$$\hom_{A\times B}((a, b), (a', b')) :\equiv \hom_A(a, a') \times \hom_B(b, b').$$

Identities are defined by $1_{(a,b)} :\equiv (1_a, 1_b)$ and composition by

$$(g, g')(f, f') :\equiv ((gf), (g'f')).$$

L 9.5.3. *For precategories A, B, C, the following types are equivalent.*
(i). Functors $A \times B \to C$.
(ii). Functors $A \to C^B$.

Now for any precategory A, we have a hom-functor

$$\hom_A : A^{\mathrm{op}} \times A \to \mathcal{S}et.$$

It takes a pair $(a, b) : (A^{\mathrm{op}})_0 \times A_0 \equiv A_0 \times A_0$ to the set $\hom_A(a, b)$. For a morphism $(f, f') : \hom_{A^{\mathrm{op}}\times A}((a, b), (a', b'))$, by definition we have $f : \hom_A(a', a)$ and $f' : \hom_A(b, b')$, so we can define

$$\begin{aligned}(\hom_A)_{(a,b),(a',b')}(f, f') &:\equiv (g \mapsto (f'gf))\\ &: \hom_A(a, b) \to \hom_A(a', b').\end{aligned}$$

Functoriality is easy to check.

T 9.5.4 (The Yoneda lemma). *For any precategory A, any $a : A$, and any functor $F : \mathcal{S}et^{A^{\mathrm{op}}}$, we have an isomorphism*

$$\hom_{\mathcal{S}et^{A^{\mathrm{op}}}}(\mathbf{y}a, F) \cong Fa. \tag{9.5.5}$$

Moreover, this is natural in both a and F.

C 9.5.6. *The Yoneda embedding $\mathbf{y} : A \to \mathcal{S}et^{A^{\mathrm{op}}}$ is fully faithful.*

C 9.5.7. *If A is a category, then $\mathbf{y}_0 : A_0 \to (\mathcal{S}et^{A^{\mathrm{op}}})_0$ is an embedding. In particular, if $\mathbf{y}a = \mathbf{y}b$, then $a = b$.*

D 9.5.8. A functor $F : \mathcal{S}et^{A^{\mathrm{op}}}$ is said to be **representable** if there exists $a : A$ and an isomorphism $\mathbf{y}a \cong F$.

T 9.5.9. *If A is a category, then the type "F is representable" is a mere proposition.*

L 9.5.10. *For any precategories A and B and a functor $F : A \to B$, the following types are equivalent.*

(i). F is a left adjoint.

(ii). For each $b : B$, the functor $(a \mapsto \hom_B(Fa, b))$ from A^{op} to $\mathcal{S}et$ is representable.

C 9.5.11. *[**??**] If A is a category and $F : A \to B$, then the type "F is a left adjoint" is a mere proposition.*

9.6 Strict categories

D 9.6.1. A **strict category** is a precategory whose type of objects is a set.

E 9.6.2. Let A be a precategory and $x : A$ an object. Then there is a precategory $\mathsf{mono}(A, x)$ as follows:

- Its objects consist of an object $y : A$ and a monomorphism $m : \hom_A(y, x)$. (As usual, $m : \hom_A(y, x)$ is a **monomorphism** (or is **monic**) if $(m \circ f = m \circ g) \Rightarrow (f = g)$.)
- Its morphisms from (y, m) to (z, n) are arbitrary morphisms from y to z in A (not necessarily respecting m and n).

An equality $(y, m) = (z, n)$ of objects in $\mathsf{mono}(A, x)$ consists of an equality $p : y = z$ and an equality $p_*(m) = n$, which by **??** is equivalently an equality $m = n \circ \mathsf{idtoiso}(p)$. Since hom-sets are sets, the type of such equalities is a mere proposition. But since m and n are monomorphisms, the type of morphisms f such that $m = n \circ f$ is also a mere proposition. Thus, if A is a category, then $(y, m) = (z, n)$ is a mere proposition, and hence $\mathsf{mono}(A, x)$ is a strict category.

E 9.6.3. Let E/F be a finite Galois extension of fields, and G its Galois group. Then there is a strict category whose objects are intermediate fields $F \subseteq K \subseteq E$, and whose morphisms are field homomorphisms which fix F pointwise (but need not commute with the inclusions into E). There is another strict category whose objects are subgroups $H \subseteq G$, and whose morphisms are morphisms of G-sets $G/H \to G/K$. The fundamental theorem of Galois theory says that these two precategories are isomorphic (not merely equivalent).

9.7 †-categories

D 9.7.1. A **†-precategory** is a precategory A together with the following.

(i). For each $x, y : A$, a function $(-)^\dagger : \hom_A(x, y) \to \hom_A(y, x)$.

(ii). For all $x : A$, we have $(1_x)^\dagger = 1_x$.

(iii). For all f, g we have $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$.

(iv). For all f we have $(f^\dagger)^\dagger = f$.

D 9.7.2. A morphism $f : \hom_A(x, y)$ in a †-precategory is **unitary** if $f^\dagger \circ f = 1_x$ and $f \circ f^\dagger = 1_y$.

L 9.7.3. *If $p : (x = y)$, then $\mathsf{idtoiso}(p)$ is unitary.*

D 9.7.4. A **†-category** is a †-precategory such that for all $x, y : A$, the function

$$(x = y) \to (x \cong^\dagger y)$$

from **??** is an equivalence.

E 9.7.5. The category $\mathcal{R}el$ from **??** becomes a †-precategory if we define $(R^\dagger)(y, x) :\equiv R(x, y)$. The proof that $\mathcal{R}el$ is a category actually shows that every isomorphism is unitary; hence $\mathcal{R}el$ is also a †-category.

E 9.7.6. Any groupoid becomes a †-category if we define $f^\dagger :\equiv f^{-1}$.

E 9.7.7. Let $\mathcal{H}ilb$ be the following precategory.

- Its objects are finite-dimensional vector spaces equipped with an inner product $\langle -, - \rangle$.
- Its morphisms are arbitrary linear maps.

By standard linear algebra, any linear map $f : V \to W$ between finite dimensional inner product spaces has a uniquely defined adjoint $f^\dagger : W \to V$, characterized by $\langle fv, w \rangle = \langle v, f^\dagger w \rangle$. In this way, $\mathcal{H}ilb$ becomes a †-precategory. Moreover, a linear isomorphism is unitary precisely when it is an **isometry**, i.e. $\langle fv, fw \rangle = \langle v, w \rangle$. It follows from this that $\mathcal{H}ilb$ is a †-category, though it is not a category (not every linear isomorphism is unitary).

9.8 The structure identity principle

D 9.8.1. A **notion of structure** (P, H) over X consists of the following.

(i). A type family $P : X_0 \to \mathcal{U}$. For each $x : X_0$ the elements of Px are called (P, H)**-structures** on x.

(ii). For $x, y : X_0$ and $\alpha : Px,\ \beta : Py$, to each $f : \hom_X(x, y)$ a mere proposition

$$H_{\alpha\beta}(f).$$

If $H_{\alpha\beta}(f)$ is true, we say that f is a (P, H)**-homomorphism** from α to β.

(iii). For $x : X_0$ and $\alpha : Px$, we have $H_{\alpha\alpha}(1_x)$.

(iv). For $x, y, z : X_0$ and $\alpha : Px,\ \beta : Py,\ \gamma : Pz$, if $f : \hom_X(x, y)$ and $g : \hom_X(y, z)$, we have

$$H_{\alpha\beta}(f) \to H_{\beta\gamma}(g) \to H_{\alpha\gamma}(g \circ f).$$

When (P, H) is a notion of structure, for $\alpha, \beta : Px$ we define

$$(\alpha \leq_x \beta) :\equiv H_{\alpha\beta}(1_x).$$

By (and (, this is a preorder (**??**) with Px its type of objects. We say that (P, H) is a **standard notion of structure** if this preorder is in fact a partial order, for all $x : X$.

T 9.8.2 (Structure identity principle)**.** *If X is a category and (P, H) is a standard notion of structure over X, then the precategory $\mathsf{Str}_{(P,H)}(X)$ is a category.*

E 9.8.3. Let A be a precategory and B a category. There is a precategory B^{A_0} whose objects are functions $A_0 \to B_0$, and whose set of morphisms from $F_0 : A_0 \to B_0$ to $G_0 : A_0 \to B_0$ is $\prod_{(a:A_0)} \hom_B(F_0 a, G_0 a)$. Composition and identities are inherited directly from those in B. It is easy to show that $\gamma : \hom_{B^{A_0}}(F_0, G_0)$ is an isomorphism exactly when each component γ_a is an isomorphism, so that we have $(F_0 \cong G_0) \simeq \prod_{(a:A_0)}(F_0 a \cong G_0 a)$. Moreover, the map $\mathsf{idtoiso} : (F_0 = G_0) \to (F_0 \cong G_0)$ of B^{A_0} is equal to the composite

$$(F_0 = G_0) \longrightarrow \prod_{a:A_0} (F_0 a = G_0 a) \longrightarrow \prod_{a:A_0} (F_0 a \cong G_0 a) \longrightarrow (F_0 \cong G_0)$$

in which the first map is an equivalence by function extensionality, the second because it is a dependent product of equivalences (since B is a category), and the third as remarked above. Thus, B^{A_0} is a category.
Now we define a notion of structure on B^{A_0} for which $P(F_0)$ is the type of operations $F : \prod_{(a,a':A_0)} \hom_A(a, a') \to \hom_B(F_0 a, F_0 a')$ which extend F_0 to a functor (i.e. preserve composition and identities). This is a set since each $\hom_B(-,-)$ is so. Given such F and G, we define $\gamma : \hom_{B^{A_0}}(F_0, G_0)$ to be a homomorphism if it forms a natural transformation. In **??** we essentially verified that this is a notion of structure. Moreover, if F and F' are both structures on F_0 and the identity is a natural transformation from F to F', then for any $f : \hom_A(a, a')$ we have $F'f = F'f \circ 1_{F_0 a} = 1_{F_0 a} \circ Ff = Ff$. Applying function extensionality, we conclude $F = F'$. Thus, we have a *standard* notion of structure, and so by **??**, the precategory B^A is a category.

D 9.8.4.

(i). For each $\mathcal{U}$-small set x define

$$Px :\equiv P_0 x \times P_1 x.$$

Here

$$P_0 x :\equiv \prod_{\omega:\Omega_0} x^{|\omega|} \to x, \text{ and}$$

$$P_1 x :\equiv \prod_{\omega:\Omega_1} x^{|\omega|} \to \mathsf{Prop}_{\mathcal{U}},$$

(ii). For $\mathcal{U}$-small sets x, y and $\alpha : P^\omega x,\ \beta : P^\omega y,\ f : x \to y$, define

$$H_{\alpha\beta}(f) :\equiv H_{0,\alpha\beta}(f) \wedge H_{1,\alpha\beta}(f).$$

Here

$$H_{0,\alpha\beta}(f) :\equiv \forall(\omega : \Omega_0).\forall(u : x^{|\omega|}).\, f(\alpha u) = \ \beta(f \circ u), \text{ and}$$

$$H_{1,\alpha\beta}(f) :\equiv \forall(\omega : \Omega_1).\forall(u : x^{|\omega|}).\, \alpha u \to \beta(f \circ u).$$

9.9 The Rezk completion

L 9.9.1. *If A, B, C are precategories and $H : A \to B$ is an essentially surjective functor, then $(- \circ H) : C^B \to C^A$ is faithful.*

L 9.9.2. *If A, B, C are precategories and $H : A \to B$ is essentially surjective and full, then $(- \circ H) : C^B \to C^A$ is fully faithful.*

T 9.9.3. *If A, B are precategories, C is a category, and $H : A \to B$ is a weak equivalence, then $(- \circ H) : C^B \to C^A$ is an isomorphism.*

Therefore, if a precategory A admits a weak equivalence functor $A \to \widehat{A}$ into a category, then that is its "reflection" into categories: any functor from A into a category will factor essentially uniquely through $\widehat{A}$. We now give two constructions of such a weak equivalence.

T 9.9.4. *For any precategory A, there is a category $\widehat{A}$ and a weak equivalence $A \to \widehat{A}$.*

E 9.9.5. Recall from **??** that for any type X there is a pregroupoid with X as its type of objects and $\mathsf{hom}(x, y) :\equiv \|x = y\|_0$. Its Rezk completion is the *fundamental groupoid* of X. Recalling that groupoids are equivalent to 1-types, it is not hard to identify this groupoid with $\|X\|_1$.

E 9.9.6. Recall from **??** that there is a precategory whose type of objects is $\mathcal{U}$ and with $\mathsf{hom}(X, Y) :\equiv \|X \to Y\|_0$. Its Rezk completion may be called the **homotopy category of types**. Its type of objects can be identified with $\|\mathcal{U}\|_1$ (see **??**).

T 9.9.7. *A precategory C is a category if and only if for every weak equivalence of precategories $H : A \to B$, the induced functor $(- \circ H) : C^B \to C^A$ is an isomorphism of precategories.*

10.1 The category of sets

10.1.1 Limits and colimits

10.1.2 Images

L 10.1.1. *For a morphism $f : \hom_A(a, b)$ in a category A, the following are equivalent.*

(i). f *is a* ***monomorphism****: for all* $x : A$ and $g, h : \hom_A(x, a)$, *if* $f \circ g = f \circ h$ *then* $g = h$.

(ii). *(If* A has pullbacks) *the diagonal map* $a \to a \times_b a$ *is an isomorphism.*

(iii). *For all* $x : A$ *and* $k : \hom_A(x, b)$, *the type* $\sum_{(h:\hom_A(x,a))}(k = f \circ h)$ *is a mere proposition.*

(iv). *For all* $x : A$ *and* $g : \hom_A(x, a)$, *the type* $\sum_{(h:\hom_A(x,a))}(f \circ g = f \circ h)$ *is contractible.*

L 10.1.2. *A function* $f : A \to B$ *between sets is injective if and only if it is a monomorphism in* $\mathcal{S}et$.

L 10.1.3. *Let* $f, g : A \to B$ *be functions between sets* A *and* B. *The set-coequalizer* $c_{f,g} : B \to Q$ *has the property that, for any set* C *and any* $h : B \to C$ *with* $h \circ f = h \circ g$, *the type*

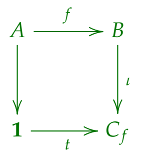
$$\sum_{k:Q\to C}(k \circ c_{f,g} = h)$$

is contractible.

L 10.1.4. *For any function* $f : A \to B$ *between sets, the following are equivalent:*

(i). f *is an epimorphism.*

(ii). *Consider the pushout diagram*



$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d] & B \arrow[d, "\iota"] \\ \mathbf{1} \arrow[r, "t"'] & C_f \end{tikzcd}$$

in $\mathcal{S}et$ *defining the mapping cone. Then the type* C_f *is contractible.*

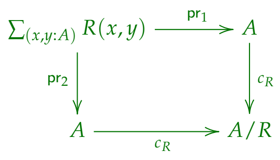
(iii). f *is surjective.*

T 10.1.5. *The category* $\mathcal{S}et$ *is regular. Moreover, surjective functions between sets are regular epimorphisms.*

L 10.1.6. *Pullbacks of regular epis in* $\mathcal{S}et$ *are regular epis.*

10.1.3 Quotients

D 10.1.7. A relation $R : A \to A \to \mathsf{Prop}$ is said to be **effective** if the square



$$\begin{tikzcd} \sum_{(x,y:A)} R(x,y) \arrow[r, "\mathsf{pr}_1"] \arrow[d, "\mathsf{pr}_2"'] & A \arrow[d, "c_R"] \\ A \arrow[r, "c_R"'] & A/R \end{tikzcd}$$

is a pullback.

L 10.1.8. *Suppose* (A, R) *is an equivalence relation. Then there is an equivalence*

$$(c_R(x) = c_R(y)) \simeq R(x, y)$$

for any $x, y : A$. *In other words, equivalence relations are effective.*

T 10.1.9. *For any function* $f : A \to B$ *between any two sets, the relation* $\mathsf{ker}(f) : A \to A \to \mathsf{Prop}$ *given by* $\mathsf{ker}(f, x, y) :\equiv (f(x) = f(y))$ *is effective.*

T 10.1.10. *Equivalence relations are effective and there is an equivalence* $A/R \simeq A /\!\!/ R$.

10.1.4 $\mathcal{S}et$ is a ΠW-pretopos

T 10.1.11. *The category $\mathcal{S}et$ is a ΠW-pretopos.*

T 10.1.12. *If there is a type $\Omega : \mathcal{U}$ of all mere propositions, then the category $\mathcal{S}et_{\mathcal{U}}$ is an elementary topos.*

10.1.5 The axiom of choice implies excluded middle

L 10.1.13. *If A is a mere proposition then its suspension $\Sigma(A)$ is a set, and A is equivalent to $\mathsf{N} =_{\Sigma(A)} \mathsf{S}$.*

T 10.1.14 (Diaconescu)**.** *The axiom of choice implies the law of excluded middle.*

T 10.1.15. *If the axiom of choice holds then the category $\mathcal{S}et$ is a well-pointed boolean elementary topos with choice.*

NB 10.1.16. The conditions on a category mentioned in the theorem are known as Lawvere's axioms for the *Elementary Theory of the Category of Sets* [**?**].

10.2 Cardinal numbers

D 10.2.1. The **type of cardinal numbers** is the 0-truncation of the type Set of sets:

$$\mathsf{Card} :\equiv \|\mathsf{Set}\|_0$$

Thus, a **cardinal number**, or **cardinal**, is an inhabitant of $\mathsf{Card} \equiv \|\mathsf{Set}\|_0$. Technically, of course, there is a separate type $\mathsf{Card}_{\mathcal{U}}$ associated to each universe $\mathcal{U}$.

D 10.2.2. The operation of **cardinal addition**

$$(- + -) : \mathsf{Card} \to \mathsf{Card} \to \mathsf{Card}$$

is defined by induction on truncation:

$$|A|_0 + |B|_0 :\equiv |A + B|_0.$$

D 10.2.3. Similarly, the operation of **cardinal multiplication**

$$(- \cdot -) : \mathsf{Card} \to \mathsf{Card} \to \mathsf{Card}$$

is defined by induction on truncation:

$$|A|_0 \cdot |B|_0 :\equiv |A \times B|_0$$

L 10.2.4. *Card is a commutative semiring, i.e. for $\alpha, \beta, \gamma : \mathsf{Card}$ we have the following.*

$$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$$
$$\alpha + 0 = \alpha$$
$$\alpha + \beta = \beta + \alpha$$
$$(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$$
$$\alpha \cdot 1 = \alpha$$
$$\alpha \cdot \beta = \beta \cdot \alpha$$
$$\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$$

where $0 :\equiv |\mathbf{0}|_0$ and $1 :\equiv |\mathbf{1}|_0$.

D 10.2.5. The operation of **cardinal exponentiation** is also defined by induction on truncation:

$$|A|_0{}^{|B|_0} :\equiv |B \to A|_0.$$

L 10.2.6. *For $\alpha, \beta, \gamma : \mathsf{Card}$ we have*

$$\alpha^0 = 1$$
$$1^\alpha = 1$$
$$\alpha^1 = \alpha$$
$$\alpha^{\beta+\gamma} = \alpha^\beta \cdot \alpha^\gamma$$
$$\alpha^{\beta\cdot\gamma} = (\alpha^\beta)^\gamma$$
$$(\alpha \cdot \beta)^\gamma = \alpha^\gamma \cdot \beta^\gamma$$

D 10.2.7. The relation of **cardinal inequality**

$$(- \leq -) : \mathsf{Card} \to \mathsf{Card} \to \mathsf{Prop}$$

is defined by induction on truncation:

$$|A|_0 \leq |B|_0 :\equiv \|\mathsf{inj}(A, B)\|$$

where $\mathsf{inj}(A, B)$ is the type of injections from A to B. In other words, $|A|_0 \leq |B|_0$ means that there merely exists an injection from A to B.

L 10.2.8. *Cardinal inequality is a preorder, i.e. for $\alpha, \beta : \mathsf{Card}$ we have*

$$\alpha \leq \alpha$$
$$(\alpha \leq \beta) \to (\beta \leq \gamma) \to (\alpha \leq \gamma)$$

L 10.2.9. *Consider the following statements:*

(i). There is an injection $A \to B$.
(ii). There is a surjection $B \to A$.

Then, assuming excluded middle:

- *Given $a_0 : A$, we have (→(.*
- *Therefore, if A is merely inhabited, we have (→ merely (.*
- *Assuming the axiom of choice, we have (→ merely (.*

T 10.2.10 (Schroeder–Bernstein)**.** *Assuming excluded middle, for sets A and B we have*

$$\mathsf{inj}(A, B) \to \mathsf{inj}(B, A) \to (A \cong B)$$

C 10.2.11. *Assuming excluded middle, cardinal inequality is a partial order, i.e. for $\alpha, \beta : \mathsf{Card}$ we have*

$$(\alpha \leq \beta) \to (\beta \leq \alpha) \to (\alpha = \beta).$$

T 10.2.12 (Cantor)**.** *For $A : \mathsf{Set}$, there is no surjection $A \to (A \to \mathbf{2})$.*

C 10.2.13. *Assuming excluded middle, for any $\alpha : \mathsf{Card}$, there is a cardinal β such that $\alpha \leq \beta$ and $\alpha \neq \beta$.*

10.3 Ordinal numbers

D 10.3.1. Let A be a set and

$$(- < -) : A \to A \to \mathsf{Prop}$$

a binary relation on A. We define by induction what it means for an element $a : A$ to be **accessible** by $<$:

- If b is accessible for every $b < a$, then a is accessible.

We write $\mathsf{acc}(a)$ to mean that a is accessible.

L 10.3.2. *Accessibility is a mere property.*

D 10.3.3. A binary relation $<$ on a set A is **well-founded** if every element of A is accessible.

L 10.3.4. *Well-foundedness is a mere property.*

E 10.3.5. Perhaps the most familiar well-founded relation is the usual strict ordering on $\mathbb{N}$. To show that this is well-founded, we must show that n is accessible for each $n : \mathbb{N}$. This is just the usual proof of "strong induction" from ordinary induction on $\mathbb{N}$.
Specifically, we prove by induction on $n : \mathbb{N}$ that k is accessible for all $k \leq n$. The base case is just that 0 is accessible, which is vacuously true since nothing is strictly less than 0. For the inductive step, we assume that k is accessible for all $k \leq n$, which is to say for all $k < n + 1$; hence by definition $n + 1$ is also accessible.
A different relation on $\mathbb{N}$ which is also well-founded is obtained by setting only $n < \mathsf{succ}(n)$ for all $n : \mathbb{N}$. Well-foundedness of this relation is almost exactly the ordinary induction principle of $\mathbb{N}$.

E 10.3.6. Let $A : \mathsf{Set}$ and $B : A \to \mathsf{Set}$ be a family of sets. Recall from **??** that the W-type $\mathsf{W}_{(a:A)} B(a)$ is inductively generated by the single constructor

- $\mathsf{sup} : \prod_{(a:A)} (B(a) \to \mathsf{W}_{(x:A)} B(x)) \to \mathsf{W}_{(x:A)} B(x)$

We define the relation $<$ on $\mathsf{W}_{(x:A)} B(x)$ by recursion on its second argument:

- For any $a : A$ and $f : B(a) \to \mathsf{W}_{(x:A)} B(x)$, we define $w < \mathsf{sup}(a, f)$ to mean that there merely exists a $b : B(a)$ such that $w = f(b)$.

Now we prove that every $w : \mathsf{W}_{(x:A)} B(x)$ is accessible for this relation, using the usual induction principle for $\mathsf{W}_{(x:A)} B(x)$. This means we assume given $a : A$ and $f : B(a) \to \mathsf{W}_{(x:A)} B(x)$, and also a lifting $f' : \prod_{(b:B(a))} \mathsf{acc}(f(b))$. But then by definition of $<$, we have $\mathsf{acc}(w)$ for all $w < \mathsf{sup}(a, f)$; hence $\mathsf{sup}(a, f)$ is accessible.

L 10.3.7. *Suppose B is a set and we have a function*

$$g : \mathcal{P}(B) \to B$$

Then if $<$ is a well-founded relation on A, there is a function $f : A \to B$ such that for all $a : A$ we have

$$f(a) = g\Big(\{\, f(a') \mid a' < a \,\}\Big).$$

L 10.3.8. *Assuming excluded middle, $<$ is well-founded if and only if every nonempty subset $B : \mathcal{P}(A)$ merely has a minimal element.*

D 10.3.9. A well-founded relation $<$ on a set A is **extensional** if for any $a, b : A$, we have

$$\Big(\forall (c : A).\, (c < a) \Leftrightarrow (c < b)\Big) \to (a = b).$$

T 10.3.10. *The type of extensional well-founded relations is a set.*

D 10.3.11. If $(A, <)$ and $(B, <)$ are extensional and well-founded, a **simulation** is a function $f : A \to B$ such that

(i). if $a < a'$, then $f(a) < f(a')$, and
(ii). for all $a : A$ and $b : B$, if $b < f(a)$, then there merely exists an $a' < a$ with $f(a') = b$.

L 10.3.12. *Any simulation is injective.*

C 10.3.13. *If $f : A \to B$ is a simulation, then for all $a : A$ and $b : B$, if $b < f(a)$, there* purely *exists an $a' < a$ with $f(a') = b$.*

T 10.3.14. *For a set A, let $P(A)$ be the type of extensional well-founded relations on A. If $<_A : P(A)$ and $<_B : P(B)$ and $f : A \to B$, let $H_{<_A <_B}(f)$ be the mere proposition that f is a simulation. Then (P, H) is a standard notion of structure over $\mathcal{S}et$ in the sense of* **??**.

C 10.3.15. *There is a category whose objects are sets equipped with extensional well-founded relations, and whose morphisms are simulations.*

L 10.3.16. *For extensional and well-founded $(A, <)$ and $(B, <)$, there is at most one simulation $f : A \to B$.*

D 10.3.17. An **ordinal** is a set A with an extensional well-founded relation which is *transitive*, i.e. satisfies $\forall (a, b, c : A).\, (a < b) \to (b < c) \to (a < c)$.

E 10.3.18. Of course, the usual strict order on $\mathbb{N}$ is transitive. It is easily seen to be extensional as well; thus it is an ordinal. As usual, we denote this ordinal by ω.

D 10.3.19. For ordinals A and B, a simulation $f : A \to B$ is said to be **bounded** if there exists $b : B$ such that $A = B_{/b}$.

T 10.3.20. *$(\mathsf{Ord}, <)$ is an ordinal.*

L 10.3.21. *Let $\mathcal{U}$ be a universe. For any $A : \mathsf{Ord}_{\mathcal{U}}$, there is a $B : \mathsf{Ord}_{\mathcal{U}}$ such that $A < B$.*

L 10.3.22. *Let $\mathcal{U}$ be a universe. For any $X : \mathcal{U}$ and $F : X \to \mathsf{Ord}_{\mathcal{U}}$, there exists $B : \mathsf{Ord}_{\mathcal{U}}$ such that $Fx \leq B$ for all $x : X$.*

10.4 Classical well-orderings

L 10.4.1. *Assuming excluded middle, every ordinal is trichotomous:*

$$\forall (a, b : A).\, (a < b) \vee (a = b) \vee (b < a).$$

L 10.4.2. *A well-founded relation contains no cycles, i.e.*

$$\forall (n : \mathbb{N}).\, \forall (a : \mathbb{N}_n \to A).\, \neg\Big((a_0 < a_1) \wedge \cdots \wedge (a_{n-1} < a_n) \wedge (a_n < a_0)\Big).$$

T 10.4.3. *Assuming excluded middle, $(A, <)$ is an ordinal if and only if every nonempty subset $B \subseteq A$ has a least element.*

T 10.4.4. *Assuming excluded middle, the following are equivalent.*

(i). *For every set X, there merely exists a function $f : \mathcal{P}_+(X) \to X$ such that $f(Y) \in Y$ for all $Y : \mathcal{P}_+(X)$.*
(ii). *Every set merely admits the structure of an ordinal.*

NB 10.4.5. If we had given the wrong proof of **??** or **??**, then the resulting proof of **??** would be invalid: there would be no way to consistently assign universe levels. As it is, we require propositional resizing (which follows from LEM) to ensure that X' lives in the same universe as X (up to equivalence).

C 10.4.6. *Assuming the axiom of choice, the function $\mathsf{Ord} \to \mathsf{Set}$ (which forgets the order structure) is a surjection.*

C 10.4.7. *Assuming AC, $\mathcal{S}et$ admits a weak equivalence functor from a strict category.*

T 10.4.8. *Assuming AC, the surjection $\mathsf{Ord} \to \mathsf{Card}$ has a section.*

10.5 The cumulative hierarchy

D 10.5.1. The **cumulative hierarchy** V relative to a type universe $\mathcal{U}$ is the higher inductive type generated by the following constructors.

(i). For every $A : \mathcal{U}$ and $f : A \to V$, there is an element $\mathsf{set}(A, f) : V$.
(ii). For all $A, B : \mathcal{U}$, $f : A \to V$ and $g : B \to V$ such that

$$(\forall(a : A). \exists(b : B). f(a) =_V g(b)) \wedge (\forall(b : B). \exists(a : A). f(a) =_V g(b)) \tag{10.5.2}$$

there is a path $\mathsf{set}(A, f) =_V \mathsf{set}(B, g)$.
(iii). The 0-truncation constructor: for all $x, y : V$ and $p, q : x = y$, we have $p = q$.

D 10.5.3. Define the **bisimulation** relation

$$\sim : V \times V \longrightarrow \mathsf{Prop}_{\mathcal{U}}$$

by double induction over V, where for $\mathsf{set}(A, f)$ and $\mathsf{set}(B, g)$ we let:

$$\mathsf{set}(A, f) \sim \mathsf{set}(B, g) :\equiv (\forall(a : A). \exists(b : B). f(a) \sim g(b)) \wedge (\forall(b : B). \exists(a : A). f(a) \sim g(b)).$$

L 10.5.4. *For any $u, v : V$ we have $(u =_V v) = (u \sim v)$.*

L 10.5.5. *For every $u : V$ there is a given $A_u : \mathcal{U}$ and monic $m_u : A_u \rightarrowtail V$ such that $u = \mathsf{set}(A_u, m_u)$.*

D 10.5.6. For $u : V$, the just constructed monic presentation $m_u : A_u \rightarrowtail V$ such that $u = \mathsf{set}(A_u, m_u)$ may be called the **type of members** of u and denoted $m_u : [u] \rightarrowtail V$, or even $[u] \rightarrowtail V$. We can think of $[u]$ as the "subclass of V consisting of members of u".

T 10.5.7. *The following hold for $(V, \in)$:*

(i). extensionality:

$$\forall(x, y : V). x \subseteq y \wedge y \subseteq x \Leftrightarrow x = y.$$

(ii). empty set: *for all $x : V$, we have $\neg(x \in \emptyset)$.*
(iii). pairing: *for all $u, v : V$, the class $\{u, v\} :\equiv \{ x \mid x = u \vee x = v \}$ is a V-set.*
(iv). infinity: *there is a $v : V$ with $\emptyset \in v$ and $x \in v$ implies $x \cup \{x\} \in v$.*
(v). union: *for all $v : V$, the class $\cup v :\equiv \{ x \mid \exists(u : V). x \in u \in v \}$ is a V-set.*
(vi). function set: *for all $u, v : V$, the class $v^u :\equiv \{ x \mid x : u \to v \}$ is a V-set.*[1]
(vii). $\in$-induction: *if $\mathsf{C} : V \to \mathsf{Prop}$ is a class such that $\mathsf{C}(a)$ holds whenever $\mathsf{C}(x)$ for all $x \in a$, then $\mathsf{C}(v)$ for all $v : V$.*

[1]Here $x : u \to v$ means that x is an appropriate set of ordered pairs, according to the usual way of encoding functions in set theory.

(viii). replacement: *given any $r : V \to V$ and $x : V$, the class*

$$\{ y \mid \exists(z : V). z \in x \land y = r(z) \}$$

is a V-set.

(ix). separation: *given any $a : V$ and $\mathcal{U}$-small $C : V \to \mathsf{Prop}_{\mathcal{U}}$, the class*

$$\{ x \mid x \in a \land C(x) \}$$

is a V-set.

C 10.5.8. *If the class $C : V \to \mathsf{Prop}$ is Δ_0 in the above sense, then it is separable.*

L 10.5.9. *In type theory with AC, the law of* ***(full) separation*** *holds for V: given* any *class $C : V \to \mathsf{Prop}$ and $a : V$, the class $a \cap C$ is a V-set.*

T 10.5.10. *In type theory with AC and a universe $\mathcal{U}$, the cumulative hierarchy V is a model of Zermelo–Fraenkel set theory with choice, ZFC.*

Notes

Exercises

ex Following the pattern of $\mathcal{S}et$, we would like to make a category $\mathcal{T}ype$ of all types and maps between them (in a given universe $\mathcal{U}$). In order for this to be a category in the sense of **??**, however, we must first declare $\mathrm{hom}(X,Y) :\equiv \|X \to Y\|_0$, with composition defined by induction on truncation from ordinary composition $(Y \to Z) \to (X \to Y) \to (X \to Z)$. This was defined as the *homotopy precategory of types* in **??**. It is still not a category, however, but only a precategory (its type of objects $\mathcal{U}$ is not even a 0-type). It becomes a category by Rezk completion (see **??**), and its type of objects can be identified with $\|\mathcal{U}\|_1$ by **??**. Show that the resulting category $\mathcal{T}ype$, unlike $\mathcal{S}et$, is not a pretopos.

Exer. 10.91. Show that if every surjection has a section in the category $\mathcal{S}et$, then the axiom of choice holds.

Exer. 10.92. Show that with LEM, the category $\mathcal{S}et$ is well-pointed, in the sense that the following statement holds: for any $f, g : A \to B$, if $f \neq g$ then there is a function $a : 1 \to A$ such that $f(a) \neq g(a)$. Show that the slice category $\mathcal{S}et/\mathbf{2}$ consisting of functions $A \to \mathbf{2}$ and commutative triangles does not have this property. (Hint: the terminal object in $\mathcal{S}et/\mathbf{2}$ is the identity function $\mathbf{2} \to \mathbf{2}$, so in this category, there are objects X that have no elements $1 \to X$.)

Exer. 10.93. Prove that if $(A, <_A)$ and $(B, <_B)$ are well-founded, extensional, or ordinals, then so is $A + B$, with $<$ defined by

$$\begin{aligned}
(a < a') &:\equiv (a <_A a') && \text{for } a, a' : A\\
(b < b') &:\equiv (b <_B b') && \text{for } b, b' : B\\
(a < b) &:\equiv \mathbf{1} && \text{for } (a : A), (b : B)\\
(b < a) &:\equiv \mathbf{0} && \text{for } (a : A), (b : B).
\end{aligned}$$

Exer. 10.94. Prove that if $(A, <_A)$ and $(B, <_B)$ are well-founded, extensional, or ordinals, then so is $A \times B$, with $<$ defined by

$$((a,b) < (a',b')) :\equiv (a <_A a') \lor ((a = a') \land (b <_B b')).$$

Exer. 10.95. Define the usual algebraic operations on ordinals, and prove that they satisfy the usual properties.

Exer. 10.96. Note that $\mathbf{2}$ is an ordinal, under the obvious relation $<$ such that $0_\mathbf{2} < 1_\mathbf{2}$ only.

(i). Define a relation $<$ on Prop which makes it into an ordinal.

(ii). Show that $\mathbf{2} =_{\mathsf{Ord}} \mathsf{Prop}$ if and only if LEM holds.

Exer. 10.97. Recall that we denote $\mathbb{N}$ by ω when regarding it as an ordinal; thus we have also the ordinal $\omega + 1$. On the other hand, let us define

$$\mathbb{N}_\infty :\equiv \left\{ a : \mathbb{N} \to \mathbf{2} \;\middle|\; \forall(n : \mathbb{N}). (a_n \le a_{\mathsf{succ}(n)}) \right\}$$

where $\le$ denotes the obvious partial order on $\mathbf{2}$, with $0_\mathbf{2} \le 1_\mathbf{2}$.

(i). Define a relation $<$ on $\mathbb{N}_\infty$ which makes it into an ordinal.

(ii). Show that $\omega + 1 =_{\mathsf{Ord}} \mathbb{N}_\infty$ if and only if the limited principle of omniscience (1) holds.

Exer. 10.98. Show that if $(A, <)$ is well-founded and extensional and $A : \mathcal{U}$, then there is a simulation $A \to V$, where $(V, \in)$ is the cumulative hierarchy from **??** built from the universe $\mathcal{U}$.

Exer. 10.99. Show that **??**(is equivalent to the axiom of choice (3).

Exer. 10.100. Given types A and B, define a **bitotal relation** to be $R : A \to B \to \mathsf{Prop}$ such that

$$\Big(\forall(a : A). \exists(b : B). R(a,b)\Big) \wedge \Big(\forall(b : B). \exists(a : A). R(a,b)\Big).$$

For such A, B, R, let $A \sqcup^R B$ be the higher inductive type generated by

- $i : A \to A \sqcup^R B$
- $j : B \to A \sqcup^R B$
- For each $a : A$ and $b : B$ such that $R(a,b)$, a path $i(a) = j(b)$.

Show that the cumulative hierarchy V can be defined by the following more straightforward list of constructors, and that the resulting induction principle is the one given in **??**.

- For every $A : \mathcal{U}$ and $f : A \to V$, there is an element $\mathsf{set}(A,f) : V$.
- For any $A, B : \mathcal{U}$ and bitotal relation $R : A \to B \to \mathsf{Prop}$, and any map $h : A \sqcup^R B \to V$, there is a path $\mathsf{set}(A, h \circ i) = \mathsf{set}(B, h \circ j)$.
- The 0-truncation constructor.

Exer. 10.101. In Constructive Zermelo–Fraenkel Set Theory, the **axiom of strong collection** has the form:

$$\Big(\forall(x \in v). \exists(y). R(x,y)\Big) \Rightarrow \\ \exists(w). \left[(\forall(x \in v). \exists(y \in w). R(x,y)) \wedge (\forall(y \in w). \exists(x \in v). R(x,y))\right]$$

Does it hold in the cumulative hierarchy V? (We do not know the answer to this.)

Exer. 10.102. Verify that, if we assume AC, then the cumulative hierarchy V satisfies the usual set-theoretic axiom of choice, which may be stated in the form:

$$\forall(x : V). \Big((\forall(y \in x). \exists(z : V). z \in y) \Rightarrow \exists(c \in (\cup x)^x). \forall(y \in x). c(y) \in y\Big)$$

Exer. 10.103. Assuming propositional resizing, show that there is a mere predicate $\mathsf{isPlump} : \mathsf{Ord} \to \mathsf{Prop}$ such that for any $A : \mathsf{Ord}$ we have

$$\mathsf{isPlump}(A) = \Big(\forall(B < A). \mathsf{isPlump}(B)\Big) \wedge \\ \Big(\forall(C, B : \mathsf{Ord}). C \leq B < A \wedge \mathsf{isPlump}(C) \Rightarrow C < A\Big).$$

Note that $\mathsf{isPlump}$ cannot be defined by a simple well-founded induction over Ord; you must use a different well-founded relation. We say that an ordinal A is **plump** [?, ?] if $\mathsf{isPlump}(A)$.

Exer. 10.104. Show that LEM is equivalent to the statement "all ordinals are plump".

Exer. 10.105. Define the **plump successor** of an ordinal A to be

$$t(A) :\equiv \{\, B : \mathsf{Ord} \mid (B \leq A) \wedge \mathsf{isPlump}(B) \,\}$$

(i). By definition, $t(A)$ belongs to the next higher universe. Show that assuming propositional resizing, it is equal to an ordinal in the same universe as A.
(ii). Again assuming propositional resizing, show that if A is plump (**??**) then so is $t(A)$.

Exer. 10.106. A **ZF-algebra** [?] relative to a universe $\mathcal{U}_i$ is a poset (see **??**) $V : \mathcal{U}_{i+1}$, which has all suprema indexed by types in $\mathcal{U}_i$, and is equipped with a "successor" function $s : V \to V$ (not necessarily respecting $\leq$ in any way).

(i). Show that the cumulative hierarchy $(V_{\mathcal{U}_i}, \subseteq, s)$ is the initial ZF-algebra, where $s(x)$ is the singleton $\{\, x \,\}$.
(ii). Show that $(\mathsf{Ord}_{\mathcal{U}_i}, \leq, s)$ is the initial ZF-algebra with the property that $x \leq s(x)$ for all x, where $s(A) = A + \mathbf{1}$ is the successor from **??**.
(iii). Assuming propositional resizing, show that $\Big(\{\, A : \mathsf{Ord}_{\mathcal{U}_i} \mid \mathsf{isPlump}(A) \,\}, \leq, t\Big)$ is the initial ZF-algebra with the property that $(x \leq y) \Rightarrow (t(x) \leq t(y))$ for all x, y, where t is the plump successor from **??**.

Exer. 10.107. For a category A, a morphism $f : \hom_A(a,b)$ is said to be a **split monomorphism** if there exists a morphism $g : \hom_A(b,a)$ such that $g \circ f = 1_a$. (Such g is called a **retraction** of f.) Prove that the following are logically equivalent.

(i). LEM.
(ii). For every sets A and B, if A is inhabited then for every monomorphism $f : A \to B$ in $\mathcal{S}et$, f is also a split monomorphism in $\mathcal{S}et$.

Homotopy Type Theory

Formal type theory

A.1 The first presentation

Convertibility $t \downarrow t'$ between terms t and t' is the equivalence relation generated by the defining equations for constants, the computation rule

A.1.1 Type universes

We postulate a hierarchy of **universes** denoted by primitive constants

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \ldots$$

The first two rules for universes say that they form a cumulative hierarchy of types:

- $\mathcal{U}_m : \mathcal{U}_n$ for $m < n$,
- if $A : \mathcal{U}_m$ and $m \leq n$, then $A : \mathcal{U}_n$,

and the third expresses the idea that an object of a universe can serve as a type and stand to the right of a colon in judgments:

- if $\Gamma \vdash A : \mathcal{U}_n$, and x is a new variable,[2] then $\vdash (\Gamma, x : A)$ ctx.

In the body of the book, an equality judgment $A \equiv B : \mathcal{U}_n$ between types A and B is usually abbreviated to $A \equiv B$. This is an instance of typical ambiguity, as we can always switch to a larger universe, which however does not affect the validity of the judgment.

The following conversion rule allows us to replace a type by one equal to it in a typing judgment:

- if $a : A$ and $A \equiv B$ then $a : B$.

A.1.2 Dependent function types (Π-types)

We introduce a primitive constant c_Π, but write $c_\Pi(A, \lambda x. B)$ as $\prod_{(x:A)} B$. Judgments concerning such expressions and expressions of the form $\lambda x. b$ are introduced by the following rules:

- if $\Gamma \vdash A : \mathcal{U}_n$ and $\Gamma, x : A \vdash B : \mathcal{U}_n$, then $\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_n$
- if $\Gamma, x : A \vdash b : B$ then $\Gamma \vdash (\lambda x. b) : (\prod_{(x:A)} B)$
- if $\Gamma \vdash g : \prod_{(x:A)} B$ and $\Gamma \vdash t : A$ then $\Gamma \vdash g(t) : B[t/x]$

If x does not occur freely in B, we abbreviate $\prod_{(x:A)} B$ as the non-dependent function type $A \to B$ and derive the following rule:

- if $\Gamma \vdash g : A \to B$ and $\Gamma \vdash t : A$ then $\Gamma \vdash g(t) : B$

Using non-dependent function types and leaving implicit the context Γ, the rules above can be written in the following alternative style that we use in the rest of this section of the appendix:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\prod_{(x:A)} B(x) : \mathcal{U}_n$
- if $x : A \vdash b : B$ then $\lambda x. b : \prod_{(x:A)} B(x)$
- if $g : \prod_{(x:A)} B(x)$ and $t : A$ then $g(t) : B(t)$

A.1.3 Dependent pair types (Σ-types)

We introduce primitive constants c_Σ and c_{pair}. An expression of the form $c_\Sigma(A, \lambda a. B)$ is written as $\sum_{(a:A)} B$, and an expression of the form $c_{\mathsf{pair}}(a, b)$ is written as (a, b). We write $A \times B$ instead of $\sum_{(x:A)} B$ if x is not free in B.

Judgments concerning such expressions are introduced by the following rules:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\sum_{(x:A)} B(x) : \mathcal{U}_n$
- if, in addition, $a : A$ and $b : B(a)$, then $(a, b) : \sum_{(x:A)} B(x)$

If we have A and B as above, $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}_m$, and

$$d : \prod_{(x:A)} \prod_{(y:B(x))} C((x, y))$$

we can introduce a defined constant

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

with the defining equation

$$f((x, y)) :\equiv d(x, y).$$

Note that C, d, x, and y may contain extra implicit parameters $x_1, \ldots, x_n$ if they were obtained in some non-empty context; therefore, the fully explicit recursion schema is

$$f(x_1, \ldots, x_n, (x(x_1, \ldots, x_n), y(x_1, \ldots, x_n))) :\equiv$$
$$d(x_1, \ldots, x_n, (x(x_1, \ldots, x_n), y(x_1, \ldots, x_n))).$$

[2]By "new" we mean that it does not appear in Γ or A.

A.1.4 Coproduct types

We introduce primitive constants c_+, c_{inl}, and c_{inr}. We write $A+B$ instead of $c_+(A,B)$, $\mathsf{inl}(a)$ instead of $c_{\mathsf{inl}}(a)$, and $\mathsf{inr}(a)$ instead of $c_{\mathsf{inr}}(a)$:

- if $A, B : \mathcal{U}_n$ then $A+B : \mathcal{U}_n$
- moreover, $\mathsf{inl} : A \to A+B$ and $\mathsf{inr} : B \to A+B$

If we have A and B as above, $C : A+B \to \mathcal{U}_m$, $d : \prod_{(x:A)} C(\mathsf{inl}(x))$, and $e : \prod_{(y:B)} C(\mathsf{inr}(y))$, then we can introduce a defined constant $f : \prod_{(z:A+B)} C(z)$ with the defining equations

$$f(\mathsf{inl}(x)) :\equiv d(x) \qquad \text{and} \qquad f(\mathsf{inr}(y)) :\equiv e(y).$$

A.1.5 The finite types

We introduce primitive constants $\star$, $\mathbf{0}$, $\mathbf{1}$, satisfying the following rules:

- $\mathbf{0} : \mathcal{U}_0$, $\mathbf{1} : \mathcal{U}_0$
- $\star : \mathbf{1}$

Given $C : \mathbf{0} \to \mathcal{U}_n$ we can introduce a defined constant $f : \prod_{(x:\mathbf{0})} C(x)$, with no defining equations.
Given $C : \mathbf{1} \to \mathcal{U}_n$ and $d : C(\star)$ we can introduce a defined constant $f : \prod_{(x:\mathbf{1})} C(x)$, with defining equation $f(\star) :\equiv d$.

A.1.6 Natural numbers

The type of natural numbers is obtained by introducing primitive constants $\mathbb{N}$, 0, and succ with the following rules:

- $\mathbb{N} : \mathcal{U}_0$,
- $0 : \mathbb{N}$,
- $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$.

Furthermore, we can define functions by primitive recursion. If we have $C : \mathbb{N} \to \mathcal{U}_k$ we can introduce a defined constant $f : \prod_{(x:\mathbb{N})} C(x)$ whenever we have

$$d : C(0)$$
$$e : \textstyle\prod_{(x:\mathbb{N})} (C(x) \to C(\mathsf{succ}(x)))$$

with the defining equations

$$f(0) :\equiv d \qquad \text{and} \qquad f(\mathsf{succ}(x)) :\equiv e(x, f(x)).$$

A.1.7 W-types

For W-types we introduce primitive constants c_{W} and c_{sup}. An expression of the form $c_{\mathsf{W}}(A, \lambda x.\, B)$ is written as $\mathsf{W}_{(x:A)} B$, and an expression of the form $c_{\mathsf{sup}}(x,u)$ is written as $\mathsf{sup}(x,u)$:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\mathsf{W}_{(x:A)} B(x) : \mathcal{U}_n$
- if moreover, $a : A$ and $u : B(a) \to \mathsf{W}_{(x:A)} B(x)$ then $\mathsf{sup}(a,u) : \mathsf{W}_{(x:A)} B(x)$.

Here also we can define functions by total recursion. If we have A and B as above and $C : (\mathsf{W}_{(x:A)} B(x)) \to \mathcal{U}_m$, then we can introduce a defined constant $f : \prod_{(z:\mathsf{W}_{(x:A)} B(x))} C(z)$ whenever we have

$$d : \textstyle\prod_{(a:A)} \prod_{(u:B(a) \to \mathsf{W}_{(x:A)} B(x))} ((\prod_{(y:B(a))} C(u(y))) \to C(\mathsf{sup}(a,u)))$$

with the defining equation

$$f(\mathsf{sup}(a,u)) :\equiv d(a, u, f \circ u).$$

A.1.8 Identity types

We introduce primitive constants $c_=$ and c_{refl}. We write $a =_A b$ for $c_=(A,a,b)$ and refl_a for $c_{\mathsf{refl}}(A,a)$, when $a : A$ is understood:

- If $A : \mathcal{U}_n$, $a : A$, and $b : A$ then $a =_A b : \mathcal{U}_n$.
- If $a : A$ then $\mathsf{refl}_a : a =_A a$.

Given $a : A$, if $y : A, z : a =_A y \vdash C : \mathcal{U}_m$ and $\vdash d : C[a, \mathsf{refl}_a / y, z]$ then we can introduce a defined constant

$$f : \textstyle\prod_{(y:A)} \prod_{(z:a =_A y)} C$$

with defining equation

$$f(a, \mathsf{refl}_a) :\equiv d.$$

A.2 The second presentation

In this section, there are three kinds of judgments

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

which we specify by providing inference rules for deriving them. A typical **inference rule** has the form

$$\frac{\mathcal{J}_1 \qquad \dots \qquad \mathcal{J}_k}{\mathcal{J}}\ \textsc{Name}$$

It says that we may derive the **conclusion** $\mathcal{J}$, provided that we have already derived the **hypotheses** $\mathcal{J}_1, \dots, \mathcal{J}_k$.
A **derivation** of a judgment is a tree constructed from such inference rules, with the judgment at the root of the tree.

A.2.1 Contexts

A context is a list

$$x_1{:}A_1, x_2{:}A_2, \dots, x_n{:}A_n$$

The judgment $\Gamma\ \mathsf{ctx}$ formally expresses the fact that Γ is a well-formed context, and is governed by the rules of inference

$$\frac{}{\cdot\ \mathsf{ctx}}\ \mathsf{ctx}\text{-}\textsc{emp} \qquad\qquad \frac{x_1{:}A_1, \dots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \dots, x_n{:}A_n)\ \mathsf{ctx}}\ \mathsf{ctx}\text{-}\textsc{ext}$$

A.2.2 Structural rules

$$\frac{(x_1{:}A_1, \dots, x_n{:}A_n)\ \mathsf{ctx}}{x_1{:}A_1, \dots, x_n{:}A_n \vdash x_i : A_i}\ \mathsf{Vble}$$

The following important principles, called **substitution** and **weakening**, need not be explicitly assumed. For the typing judgments these principles are manifested as

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b : B}{\Gamma, \Delta[a/x] \vdash b[a/x] : B[a/x]}\ \mathsf{Subst}_1 \qquad\qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b : B}{\Gamma, x{:}A, \Delta \vdash b : B}\ \mathsf{Wkg}_1$$

and for judgmental equalities they become

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b \equiv c : B}{\Gamma, \Delta[a/x] \vdash b[a/x] \equiv c[a/x] : B[a/x]}\ \mathsf{Subst}_2 \qquad\qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b \equiv c : B}{\Gamma, x{:}A, \Delta \vdash b \equiv c : B}\ \mathsf{Wkg}_2$$

In addition to the judgmental equality rules given for each type former, we also assume that judgmental equality is an equivalence relation respected by typing.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

Additionally, for all the type formers below, we assume rules stating that each constructor preserves definitional equality in each of its arguments; for instance, along with the Π-INTRO rule, we assume the rule

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash b \equiv b' : B}{\Gamma \vdash \lambda x.\, b \equiv \lambda x.\, b' : \prod_{(x:A)} B}\ \Pi\text{-}\textsc{intro-eq}$$

However, we omit these rules for brevity.

A.2.3 Type universes

We postulate an infinite hierarchy of type universes

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \dots$$

Each universe is contained in the next, and any type in $\mathcal{U}_i$ is also in $\mathcal{U}_{i+1}$:

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-}\textsc{intro} \qquad\qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-}\textsc{cumul}$$

A.2.4 Dependent function types (Π-types)

$$x{:}A \vdash B : \mathcal{U}_i.$$

- a **formation rule**, stating when the type former can be applied;
- some **introduction rules**, stating how to inhabit the type;
- **elimination rules**, or an induction principle, stating how to use an element of the type;
- **computation rules**, which are judgmental equalities explaining what happens when elimination rules are applied to results of introduction rules;
- optional **uniqueness principles**, which are judgmental equalities explaining how every element of the type is uniquely determined by the results of elimination rules applied to it.

For the dependent function type these rules are:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-}\textsc{form} \qquad \frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda(x:A).\, b : \prod_{(x:A)} B}\ \Pi\text{-}\textsc{intro} \qquad \frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-}\textsc{elim} \qquad \frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-}\textsc{comp}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda x.\, f(x)) : \prod_{(x:A)} B}\ \Pi\text{-}\textsc{uniq}$$

A.2.5 Dependent pair types (Σ-types)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}_i}\ \Sigma\text{-}\textsc{form} \qquad \frac{\Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : \sum_{(x:A)} B}\ \Sigma\text{-}\textsc{intro} \qquad \frac{\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \qquad \Gamma \vdash p : \sum_{(x:A)} B}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, p) : C[p/z]}\ \Sigma\text{-}\textsc{elim}$$

$$\frac{\begin{array}{c}\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, (a,b)) \equiv g[a,b/x,y] : C[(a,b)/z]}\ \Sigma\text{-}\textsc{comp}$$

A.2.6 Coproduct types

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A + B : \mathcal{U}_i}\ +\text{-}\textsc{form}$$

$$\frac{\begin{array}{c}\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{inl}(a) : A + B}\ +\text{-}\textsc{intro}_1 \qquad \frac{\begin{array}{c}\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \\ \Gamma \vdash b : B\end{array}}{\Gamma \vdash \mathsf{inr}(b) : A + B}\ +\text{-}\textsc{intro}_2$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash e : A + B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, e) : C[e/z]}\ +\text{-}\textsc{elim} \qquad \frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inl}(a)) \equiv c[a/x] : C[\mathsf{inl}(a)/z]}\ +\text{-}\textsc{comp}_1$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash b : B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inr}(b)) \equiv d[b/y] : C[\mathsf{inr}(b)/z]}\ +\text{-}\textsc{comp}_2$$

A.2.7 The empty type $\mathbf{0}$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_i}\ \mathbf{0}\text{-}\textsc{form} \qquad \frac{\Gamma, x{:}\mathbf{0} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}(x.C, a) : C[a/x]}\ \mathbf{0}\text{-}\textsc{elim}$$

A.2.8 The unit type $\mathbf{1}$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\ \mathbf{1}\text{-}\textsc{form} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-}\textsc{intro} \qquad \frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x] \qquad \Gamma \vdash a : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, a) : C[a/x]}\ \mathbf{1}\text{-}\textsc{elim} \qquad \frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x]}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, \star) \equiv c : C[\star/x]}\ \mathbf{1}\text{-}\textsc{comp}$$

A.2.9 The natural number type

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1 \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n) : C[n/x]}\ \mathbb{N}\text{-ELIM}$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x]}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, 0) \equiv c_0 : C[0/x]}\ \mathbb{N}\text{-COMP}_1$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, \mathsf{succ}(n)) \\ \qquad \equiv c_s[n, \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n)/x, y] : C[\mathsf{succ}(n)/x]\end{array}}\ \mathbb{N}\text{-COMP}_2$$

A.2.10 Identity types

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i}\ {=}\text{-FORM} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}\ {=}\text{-INTRO}$$

$$\frac{\Gamma, x{:}A, y{:}A, p{:}x =_A y \vdash C : \mathcal{U}_i \qquad \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z/x, y, p] \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p' : a =_A b}{\Gamma \vdash \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, b, p') : C[a, b, p'/x, y, p]}\ {=}\text{-ELIM}$$

$$\frac{\Gamma, x{:}A, y{:}A, p{:}x =_A y \vdash C : \mathcal{U}_i \qquad \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z/x, y, p] \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, a, \mathsf{refl}_a) \equiv c[a/z] : C[a, a, \mathsf{refl}_a/x, y, p]}\ {=}\text{-COMP}$$

A.3 Homotopy type theory

A.3.1 Function extensionality and univalence

?? is formalized by introduction of a constant funext which asserts that happly is an equivalence:

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash g : \prod_{(x:A)} B}{\Gamma \vdash \mathsf{funext}(f, g) : \mathsf{isequiv}(\mathsf{happly}_{f,g})}\ \Pi\text{-EXT}$$

The definitions of happly and isequiv can be found in (2) and **??**, respectively.
?? is formalized in a similar fashion, too:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash \mathsf{univalence}(A, B) : \mathsf{isequiv}(\mathsf{idtoeqv}_{A,B})}\ \mathcal{U}_i\text{-UNIV}$$

The definition of idtoeqv can be found in (2).

A.3.2 The circle

Here we give an example of a basic higher inductive type; others follow the same general scheme, albeit with elaborations.
Note that the rules below do not precisely follow the pattern of the ordinary inductive types in **??**: the rules refer to the notions of transport and functoriality of maps (**??**), and the second computation rule is a propositional, not judgmental, equality. These differences are discussed in **??**.

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{S}^1 : \mathcal{U}_i}\ \mathbb{S}^1\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{base} : \mathbb{S}^1}\ \mathbb{S}^1\text{-INTRO}_1 \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}}\ \mathbb{S}^1\text{-INTRO}_2$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b \qquad \Gamma \vdash p : \mathbb{S}^1}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, p) : C[p/x]}\ \mathbb{S}^1\text{-ELIM}$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, \mathsf{base}) \equiv b : C[\mathsf{base}/x]}\ \mathbb{S}^1\text{-COMP}_1$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathbb{S}^1\text{-}\mathsf{loopcomp} : \mathsf{apd}_{(\lambda y.\, \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, y))}(\mathsf{loop}) = \ell}\ \mathbb{S}^1\text{-COMP}_2$$

In $\mathsf{ind}_{\mathbb{S}^1}$, x is bound in C. The notation $b =^{C}_{\mathsf{loop}} b$ for dependent paths was introduced in Section 6.2.

A.4 Basic metatheory

T A.4.1. *If $A : \mathcal{U}$ and $A \downarrow A'$ then $A' : \mathcal{U}$. If $t : A$ and $t \downarrow t'$ then $t' : A$.*

T A.4.2. *If $A : \mathcal{U}$ then A is strongly normalizable. If $t : A$ then A and t are strongly normalizable.*