Homotopy Type Theory

Basics

2.1 Types are higher groupoids

Lemma 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.*

Lemma 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

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

Theorem 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 $\alpha, \beta : \Omega^2(A)$.

Definition 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 $\mathcal{U}$.

Definition 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

Lemma 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.

Lemma 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

Lemma 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.

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

Lemma 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 $\mathsf{pr}_1(\mathsf{lift}(u, p)) = p$.

Remark 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".

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

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

Lemma 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*

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

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

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

Lemma 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*

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

Lemma 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*

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

Lemma 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

Definition 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 §2.9 we will introduce an axiom making homotopies and identifications "equivalent".
The following proofs are left to the reader.

Lemma 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*

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

Lemma 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:

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

Corollary 2.4.5. *Let $H : f \sim \mathrm{id}_A$ be a homotopy, with $f : A \to A$. Then for any $x : A$ we have*
$$H(f(x)) = f(H(x)).$$

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

Definition 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, (2.4.6) is *the type of quasi-inverses of f*; we may denote it by $\mathsf{qinv}(f)$.

Example 2.4.8. For any $p : x =_A y$ and $z : A$, the functions
$$\begin{aligned}(p \cdot -) &: (y =_A z) \to (x =_A z) \qquad \text{and}\\ (- \cdot p) &: (z =_A x) \to (z =_A y)\end{aligned}$$
have quasi-inverses given by $(p^{-1} \cdot -)$ and $(- \cdot p^{-1})$, respectively;

Example 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 Lemma 2.3.9.

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

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

Lemma 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

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

Theorem 2.6.2. *For any x and y, the function (2.6.1) is an equivalence.*

Theorem 2.6.3. *In the above situation, we have*
$$\mathsf{transport}^{A\times B}(p, x) =_{A(w)\times B(w)} (\mathsf{transport}^A(p, \mathsf{pr}_1 x), \mathsf{transport}^B(p, \mathsf{pr}_2 x)).$$

Theorem 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

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

Corollary 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 Lemma 2.3.2 may be identified with the special case of the introduction form
$$\mathsf{pair}^=(p, \mathsf{refl}_{p_*(u)}) : (x, u) = (y, p_*(u)).$$

Theorem 2.7.4. *Suppose we have type families*
$$P : A \to \mathcal{U} \qquad \text{and} \qquad Q : \Big(\sum_{x:A} P(x)\Big) \to \mathcal{U}.$$
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

Theorem 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

$$\mathsf{happly} : (f = g) \to \prod_{x:A} (f(x) =_{B(x)} g(x)) \tag{2.9.1}$$

Axiom 2.9.2 (Function extensionality). *For any A, B, f, and g, the function (2.9.1) is an equivalence.*

In particular, Axiom 2.9.2 implies that (2.9.1) has a quasi-inverse
$$\mathsf{funext} : \Big(\prod_{x:A} (f(x) = g(x))\Big) \to (f = g).$$

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

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

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
$$\mathsf{transport}^{A\to B}(p, f) = \Big(x \mapsto \mathsf{transport}^B(p, f(\mathsf{transport}^A(p^{-1}, x)))\Big) \tag{2.9.3}$$
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
$$\mathsf{transport}^{\Pi_A(B)}(p, f)(a) =$$
$$\mathsf{transport}^{\widehat{B}}\Big((\mathsf{pair}^=(p^{-1}, \mathsf{refl}_{p^{-1}{}_*(a)}))^{-1}, f(\mathsf{transport}^A(p^{-1}, a))\Big)$$
where $\Pi_A(B)$ and $\widehat{B}$ denote respectively the type families
$$\begin{aligned}\Pi_A(B) &:\equiv (x \mapsto \textstyle\prod_{(a:A(x))} B(x, a)) &&: X \to \mathcal{U}\\ \widehat{B} &:\equiv (w \mapsto B(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) &&: (\textstyle\sum_{(x:X)} A(x)) \to \mathcal{U}.\end{aligned} \tag{2.9.4}$$

Lemma 2.9.5. *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 (2.9.3),*
- *$j : p_*(f(p^{-1}{}_*(p_*(a)))) = p_*(f(a))$ comes from Lemmas 2.1.3 and 2.3.9, and*
- *$k : p_*(f(a)) = g(p_*(a))$ is $\widehat{q}(a)$.*

Lemma 2.9.6. *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 (2.9.4).

2.10 Universes and the univalence axiom

Lemma 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.

Axiom 2.10.3 (Univalence). *For any $A, B : \mathcal{U}$, the function (2.10.2) 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:

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

Lemma 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

Theorem 2.11.1. *If $f : A \to B$ is an equivalence, then for all $a, a' : A$, so is*

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

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

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

Theorem 2.11.3. *For $f, g : A \to B$, with $p : a =_A a'$ and $q : f(a) =_B g(a)$, we have*

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

Theorem 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*

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

Theorem 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

Theorem 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

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

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

2.14 Example: equality of structures

Definition 2.14.1. Given a type A, the type 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 Lemmas 2.1.3 and 2.3.9.

2.15 Universal properties

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

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

Theorem 2.15.2. *(2.15.1) 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 $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$.

Theorem 2.15.4. *(2.15.3) is an equivalence.*

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

Theorem 2.15.6. *(2.15.5) is an equivalence.*

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

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

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

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

Exercise 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

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

Exercise 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 (2.15.7). 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.