

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

$$(x = y) \rightarrow (y = x)$$

denoted $p \mapsto p^{-1}$, such that $\text{refl}_x^{-1} \equiv \text{refl}_x$ 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

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

written $p \mapsto q \mapsto p \cdot q$, such that $\text{refl}_x \cdot \text{refl}_x \equiv \text{refl}_x$ 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) $p = p \cdot \text{refl}_y$ and $p = \text{refl}_x \cdot p$.
- (ii) $p^{-1} \cdot p = \text{refl}_y$ and $p \cdot p^{-1} = \text{refl}_x$.
- (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

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

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 $\mathcal{U}_\bullet := \sum_{(A, a)} A$ 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:

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

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:

$$\Omega^0(A, a) := (A, a)$$

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

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 $f : A \rightarrow B$ is a function. Then for any $x, y : A$ there is an operation

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

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

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 $f : A \rightarrow B$ and $g : B \rightarrow C$ and paths $p : x =_A y$ and $q : y =_A z$, we have:

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

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 $p_* : P(x) \rightarrow P(y)$.

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

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

Lemma 2.3.2 (Path lifting property). Let $P : A \rightarrow \mathcal{U}$ 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

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

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

Remark 2.3.3. Although we may think of a type family $P : A \rightarrow \mathcal{U}$ as like a fibration, it is generally not a good idea to say things like “the fibration $P : A \rightarrow \mathcal{U}$ ”, since this sounds like we are talking about a fibration with base \mathcal{U} and total space A . To repeat, when a type family $P : A \rightarrow \mathcal{U}$ 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

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

Lemma 2.3.5. If $P : A \rightarrow \mathcal{U}$ 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

$$\text{transportconst}_p^B(b) : \text{transport}^P(p, b) = b.$$

Lemma 2.3.6. For $f : A \rightarrow B$ and $p : x =_A y$, we have

$$\text{apd}_f(p) = \text{transportconst}_p^B(f(x)) \cdot \text{ap}_f(p).$$

Lemma 2.3.7. Given $P : A \rightarrow \mathcal{U}$ with $p : x =_A y$ and $q : y =_A z$ while $u : P(x)$, we have

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

Lemma 2.3.8. For a function $f : A \rightarrow B$ and a type family $P : B \rightarrow \mathcal{U}$, and any $p : x =_A y$ and $u : P(f(x))$, we have

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

Lemma 2.3.9. For $P, Q : A \rightarrow \mathcal{U}$ and a family of functions $f : \prod_{(x:A)} P(x) \rightarrow Q(x)$, and any $p : x =_A y$ and $u : P(x)$, we have

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

2.4 Homotopies and equivalences

Definition 2.4.1. Let $f, g : \prod_{(x:A)} P(x)$ be two sections of a type family $P : A \rightarrow \mathcal{U}$. 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

$$\begin{aligned} & \prod_{f:\prod_{(x:A)} P(x)} (f \sim f) \\ & \prod_{f,g:\prod_{(x:A)} P(x)} (f \sim g) \rightarrow (g \sim f) \\ & \prod_{f,g,h:\prod_{(x:A)} P(x)} (f \sim g) \rightarrow (g \sim h) \rightarrow (f \sim h). \end{aligned}$$

Lemma 2.4.3. Suppose $H : f \sim g$ is a homotopy between functions $f, g : A \rightarrow B$ 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:

$$\begin{array}{ccc} f(x) & \xrightarrow{f(p)} & f(y) \\ H(x) \parallel & & \parallel H(y) \\ g(x) & \xrightarrow[g(p)]{} & g(y) \end{array}$$

Corollary 2.4.4. Let $H : f \sim \text{id}_A$ be a homotopy, with $f : A \rightarrow A$. Then for any $x : A$ we have

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

$$\sum_{g:B \rightarrow A} ((f \circ g \sim \text{id}_B) \times (g \circ f \sim \text{id}_A)) \quad (2.4.5)$$

Definition 2.4.6. For a function $f : A \rightarrow B$, a **quasi-inverse** of f is a triple (g, α, β) consisting of a function $g : B \rightarrow A$ and homotopies $\alpha : f \circ g \sim \text{id}_B$ and $\beta : g \circ f \sim \text{id}_A$.

Thus, (2.4.5) is the type of quasi-inverses of f ; we may denote it by $\text{qinv}(f)$.

Example 2.4.7. For any $p : x =_A y$ and $z : A$, the functions

$$(p \bullet -) : (y =_A z) \rightarrow (x =_A z) \quad \text{and} \\ (- \bullet p) : (z =_A x) \rightarrow (z =_A y)$$

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

Example 2.4.8. For any $p : x =_A y$ and $P : A \rightarrow \mathcal{U}$, the function

$$\text{transport}^P(p, -) : P(x) \rightarrow P(y)$$

has a quasi-inverse given by $\text{transport}^P(p^{-1}, -)$; this follows from Lemma 2.3.7.

$$(A \simeq B) \equiv \sum_{f:A \rightarrow B} \text{isequiv}(f). \quad (2.4.9)$$

Lemma 2.4.10. 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

$$(x =_{A \times B} y) \rightarrow (\text{pr}_1(x) =_A \text{pr}_1(y)) \times (\text{pr}_2(x) =_B \text{pr}_2(y)). \quad (2.6.1)$$

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

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

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

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

2.7 Σ -types

Theorem 2.7.1. Suppose that $P : A \rightarrow \mathcal{U}$ is a type family over a type A and let $w, w' : \sum_{(x:A)} P(x)$. Then there is an equivalence

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

Corollary 2.7.2. For $z : \sum_{(x:A)} P(x)$, we have $z = (\text{pr}_1(z), \text{pr}_2(z))$.

Theorem 2.7.3. Suppose we have type families

$$P : A \rightarrow \mathcal{U} \quad \text{and} \quad Q : \left(\sum_{x:A} P(x) \right) \rightarrow \mathcal{U}.$$

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

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

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

$$\text{happly} : (f = g) \rightarrow \prod_{x:A} (f(x) =_{B(x)} g(x)) \quad (2.9.1)$$

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

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

This function is also referred to as “function extensionality”.

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

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

$$\text{transport}^{A \rightarrow B}(p, f) = \left(x \mapsto \text{transport}^B(p, f(\text{transport}^A(p^{-1}, x))) \right) \quad (2.9.3)$$

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

$$(A \rightarrow B)(x) \equiv (A(x) \rightarrow B(x)).$$

Transporting dependent functions is similar, but more complicated.

Suppose given X and p as before, type families $A : X \rightarrow \mathcal{U}$ and $B : \prod_{(x:X)} (A(x) \rightarrow \mathcal{U})$, and also a dependent function $f : \prod_{(a:A(x_1))} B(x_1, a)$. Then for $a : A(x_2)$, we have

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

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

$$\begin{aligned} \Pi_A(B) &\equiv (x \mapsto \prod_{(a:A(x))} B(x, a)) & : & X \rightarrow \mathcal{U} \\ \widehat{B} &\equiv (w \mapsto B(\text{pr}_1 w, \text{pr}_2 w)) & : & \left(\sum_{(x:X)} A(x) \right) \rightarrow \mathcal{U}. \end{aligned} \quad (2.9.4)$$

Lemma 2.9.5. Given type families $A, B : X \rightarrow \mathcal{U}$ and $p : x =_X y$, and also $f : A(x) \rightarrow B(x)$ and $g : A(y) \rightarrow B(y)$, 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

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

is equal to the concatenated path $i \bullet j \bullet k$, 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 2.3.7, and
- $k : p_*(f(a)) = g(p_*(a))$ is $\widehat{q}(a)$.

Lemma 2.9.6. Given type families $A : X \rightarrow \mathcal{U}$ and $B : \prod_{(x:X)} A(x) \rightarrow \mathcal{U}$ 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

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

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,

$$\text{idtoeqv} : (A =_{\mathcal{U}} B) \rightarrow (A \simeq B), \quad (2.10.2)$$

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”:

$$\text{ua} : (A \simeq B) \rightarrow (A =_{\mathcal{U}} B).$$

- The elimination rule, which is idtoeqv ,

$$\text{idtoeqv} \equiv \text{transport}^{X \rightarrow X} : (A =_{\mathcal{U}} B) \rightarrow (A \simeq B).$$

- The propositional computation rule,

$$\text{transport}^{X \mapsto X}(\text{ua}(f), x) = f(x).$$

- The propositional uniqueness principle: for any $p : A = B$,

$$p = \text{ua}(\text{transport}^{X \mapsto X}(p)).$$

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

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

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

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

2.11 Identity type

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

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

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

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

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

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

Theorem 2.11.4. Let $B : A \rightarrow \mathcal{U}$ and $f, g : \prod_{(x:A)} B(x)$, with $p : a =_A a'$ and $q : f(a) =_{B(a)} g(a)$. Then we have

$$\text{transport}^{x \mapsto f(x)=Bg(x)}(p, q) = (\text{ap}_f(p))^{-1} \bullet \text{ap}_{(\text{transport}^B p)}(q) \bullet \text{ap}_g(p).$$

Theorem 2.11.5. For $p : a =_A a'$ with $q : a = a$ and $r : a' = a'$, we have

$$(\text{transport}^{x \mapsto (x=x)}(p, q) = r) \simeq (q \bullet p = p \bullet r).$$

2.12 Coproducts

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

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.

$$\text{code} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathcal{U},$$

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

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

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

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

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

2.14 Example: equality of structures

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

$$\text{SemigroupStr}(A) := \sum_{(m:A \rightarrow A \rightarrow A)} \prod_{(x,y,z:A)} m(x, m(y, z)) = m(m(x, y), z).$$

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

$$\text{Semigroup} := \sum_{A:\mathcal{U}} \text{SemigroupStr}(A)$$

2.14.1 Lifting equivalences

$$\text{transport}^{\text{SemigroupStr}}(\text{ua}(e)) : \text{SemigroupStr}(A) \rightarrow \text{SemigroupStr}(B).$$

Moreover, this map is an equivalence, because $\text{transport}^C(\alpha)$ is always an equivalence with inverse $\text{transport}^C(\alpha^{-1})$, see Lemmas 2.1.3 and 2.3.7.

2.15 Universal properties

$$(X \rightarrow A \times B) \rightarrow (X \rightarrow A) \times (X \rightarrow B) \quad (2.15.1)$$

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

Theorem 2.15.2. (2.15.1) is an equivalence.

$$\left(\prod_{x:X} (A(x) \times B(x)) \right) \rightarrow \left(\prod_{x:X} A(x) \right) \times \left(\prod_{x:X} B(x) \right) \quad (2.15.3)$$

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

Theorem 2.15.4. (2.15.3) is an equivalence.

$$\left(\prod_{x:X} \sum_{(a:A(x))} P(x, a) \right) \rightarrow \left(\sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x)) \right). \quad (2.15.5)$$

Theorem 2.15.6. (2.15.5) is an equivalence.