

# Homotopy Type Theory

## Sets and logic

### 3.1 Sets and $n$ -types

**Definition 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  $\text{isSet}(A)$  is defined to be the type

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

*Example 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.

*Example 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  $\mathbf{0}$ .

*Example 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.

*Example 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  $p = \text{pair}^=(\text{ap}_{\text{pr}_1}(p), \text{ap}_{\text{pr}_2}(p))$  and  $q = \text{pair}^=(\text{ap}_{\text{pr}_1}(q), \text{ap}_{\text{pr}_2}(q))$ . But  $\text{ap}_{\text{pr}_1}(p) = \text{ap}_{\text{pr}_1}(q)$  since  $A$  is a set, and  $\text{ap}_{\text{pr}_2}(p) = \text{ap}_{\text{pr}_2}(q)$  since  $B$  is a set; hence  $p = q$ . Similarly, if  $A$  is a set and  $B : A \rightarrow \mathcal{U}$  is such that each  $B(x)$  is a set, then  $\sum_{(x:A)} B(x)$  is a set.

*Example 3.1.6.* If  $A$  is any type and  $B : A \rightarrow \mathcal{U}$  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

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

But for any  $x : A$ , we have

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

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

**Definition 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  $r = s$ .

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

### 3.2 Propositions as types?

*Remark 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.

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

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

*Remark 3.2.4.* For any  $A$ ,  $\neg\neg\neg A \rightarrow \neg A$  for any  $A$ .

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

### 3.3 Mere propositions

**Definition 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  $\text{isProp}(P)$  is defined to be

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

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

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

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

**Lemma 3.3.5.** For any type  $A$ , the types  $\text{isProp}(A)$  and  $\text{isSet}(A)$  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:

$$\text{LEM} := \prod_{A:\mathcal{U}} (\text{isProp}(A) \rightarrow (A + \neg A)). \quad (3.4.1)$$

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

$$\prod_{A:\mathcal{U}} (\text{isProp}(A) \rightarrow (\neg\neg A \rightarrow A)). \quad (3.4.2)$$

**Definition 3.4.3.**

- (i). A type  $A$  is called **decidable** if  $A + \neg A$ .
- (ii). Similarly, a type family  $B : A \rightarrow \mathcal{U}$  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

**Lemma 3.5.1.** Suppose  $P : A \rightarrow \mathcal{U}$  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  $\text{pr}_1(u) = \text{pr}_1(v)$ , then  $u = v$ .

For instance, recall that

$$(A \simeq B) := \sum_{f:A \rightarrow B} \text{isequiv}(f),$$

where each type  $\text{isequiv}(f)$  was supposed to be a mere proposition. It follows that if two equivalences have equal underlying functions, then they are equal as equivalences.

If  $P : A \rightarrow \mathcal{U}$  is a family of mere propositions (i.e. each  $P(x)$  is a mere proposition), we may write

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

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

$$\begin{aligned} \text{Set}_{\mathcal{U}} &:= \{A : \mathcal{U} \mid \text{isSet}(A)\}, \\ \text{Prop}_{\mathcal{U}} &:= \{A : \mathcal{U} \mid \text{isProp}(A)\}. \end{aligned}$$

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

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

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

$$\mathcal{P}(A) := (A \rightarrow \Omega),$$

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

### 3.6 The logic of mere propositions

*Example 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 Example 3.1.5 but simpler. Thus, the connective “and” preserves mere propositions.

*Example 3.6.2.* If  $A$  is any type and  $B : A \rightarrow \mathcal{U}$  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 Example 3.1.6 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  $A \rightarrow B$  regardless of what  $A$  is. In even more particular, since  $\mathbf{0}$  is a mere proposition, so is  $\neg A \equiv (A \rightarrow \mathbf{0})$ . 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  $f : A \rightarrow B$ , then there is an induced  $g : \|A\| \rightarrow B$  such that  $g(|a|) \equiv f(a)$  for all  $a : A$ .

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

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

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

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

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

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

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,

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

Of course, this is a direct translation of (3.2.1) 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

$$\left( \forall (x : X). \exists (a : A(x)). P(x, a) \right) \Rightarrow \left( \exists (g : \prod_{(x:X)} A(x)). \forall (x : X). P(x, g(x)) \right) \quad \text{Lemma 3.11.3. For any type } A, \text{ the type } \text{isContr}(A) \text{ is a mere proposition.}$$

**Lemma 3.8.2.** The axiom of choice (3.8.1) is equivalent to the statement that for any set  $X$  and any  $Y : X \rightarrow \mathcal{U}$  such that each  $Y(x)$  is a set, we have

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

**Remark 3.8.4.** The right side of (3.8.3) always implies the left. Since both are mere propositions, by Lemma 3.3.3 the axiom of choice is also equivalent to asking for an equivalence

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

**Lemma 3.8.5.** There exists a type  $X$  and a family  $Y : X \rightarrow \mathcal{U}$  such that each  $Y(x)$  is a set, but such that (3.8.3) is false.

### 3.9 The principle of unique choice

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

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

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

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

### 3.11 Contractibility

**Definition 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  $\text{contr}_x$ .

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

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

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

- $A$  is contractible in the sense of Definition 3.11.1.
- $A$  is a mere proposition, and there is a point  $a : A$ .
- $A$  is equivalent to  $\mathbf{1}$ .

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

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

**Lemma 3.11.5.** If  $P : A \rightarrow \mathcal{U}$  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  $r : A \rightarrow B$  such that there exists a function  $s : B \rightarrow A$ , called its **section**, and a homotopy  $\epsilon : \prod_{(y:B)} (r(s(y)) = y)$ ; then we say that  $B$  is a **retract** of  $A$ .

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

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

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

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

**Lemma 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.