Homotopy Type Theory

Sets and logic

3.1 Sets and n-types

defn]defn 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

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

eg]eg 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.

eg]eg 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 $\mathbf{0}$.

eg]eg 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.
eg]eg 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.

eg]eg 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.

defn]defn 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 $r = s$.

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

3.2 Propositions as types?

rmk]rmk (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.

rmk]rmkThe classical *law of double negation* and *law of excluded middle* are incompatible with the univalence axiom.

Theorem 3.2.1. *It is not the case that for all $A : \mathcal{U}$ we have $\neg(\neg A) \to A$.*

rmk]rmkFor any A, $\neg\neg\neg A \to \neg A$ for any A.

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

3.3 Mere propositions

defn]defn 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

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

lem]lem If P is a mere proposition and $x_0 : P$, then $P \simeq \mathbf{1}$.

lem]lem If P and Q are mere propositions such that $P \to Q$ and $Q \to P$, then $P \simeq Q$.

lem]lem Every mere proposition is a set.

lem]lem 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

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

defn]defn
Definition 3.4.2. (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

lem]lem 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

$$\{ x : A \mid P(x) \} \tag{3.5.1}$$

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 $\mathsf{Prop}_{\mathcal{U}}$.
axiom]axiomThe 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

eg]egIf 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.

eg]eg 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 $a : A$.

defn]defn 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,

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

Of course, this is a direct translation of (**??**) 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

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

*lem]lem The axiom of choice (**??**) 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*

$$\Big(\prod_{x:X} \big\| Y(x) \big\| \Big) \to \Big\| \prod_{x:X} Y(x) \Big\|. \tag{3.8.2}$$

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

$$\Big(\prod_{x:X} \big\| Y(x) \big\| \Big) \simeq \Big\| \prod_{x:X} Y(x) \Big\|$$

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

3.9 The principle of unique choice

lem]lem If P is a mere proposition, then $P \simeq \|P\|$.

cor]cor Suppose a type family $P : A \to \mathcal{U}$ such that

Corollary 3.9.0 (The principle of unique choice)**.** *(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

defn]defn 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

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

lem]lem For a type A, the following are logically equivalent.

Lemma 3.11.0. *(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 **1**.*

lem]lem For any type A, the type $\mathsf{isContr}(A)$ is a mere proposition.

cor]cor If A is contractible, then so is $\mathsf{isContr}(A)$.

lem]lem 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.

lem]lem If B is a retract of A, and A is contractible, then so is B.

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

lem]lem Let $P : A \to \mathcal{U}$ be a type family.

Lemma 3.11.0. *(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 $P(a)$.

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