HoTT Chapters 2.4 through 2.8

Paul Wild

Friedrich-Alexander-Universität Erlangen-Nürnberg

Notions of identification

The identity type $x =_A y$ can be regarded as a type of *identifications*, *paths*, or *equivalences* between two elements x and y of a type A.
But traditionally, we regard two functions as the same if they take equal values on all inputs. This suggests that two functions f and g (perhaps dependently typed) should be the same if the type $\prod_{(x:A)}(f(x) = g(x))$ is inhabited.
Under the homotopical interpretation, this dependent function type may be regarded as the type of *homotopies* or of *natural isomorphisms*.

Homotopy

Definition

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, later we will introduce an axiom making homotopies and identifications “equivalent”.

Homotopy properties

Lemma

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

$$\prod_{f:\prod_{(x:A)} P(x)} (f \sim f)$$

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

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

Homotopy properties (cont.)

Lemma

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:

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

Quasi-inverse

Definition

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 $\alpha : f \circ g \sim \mathrm{id}_B$ and $\beta : g \circ f \sim \mathrm{id}_A$.

Thus,

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

is *the type of quasi-inverses of* f; we may denote it by $\mathrm{qinv}(f)$.
A homotopical perspective suggests that this should be called a *homotopy equivalence*, and from a categorical one, it should be called an *equivalence of (higher) groupoids*.

Equivalence

When doing proof-relevant mathematics, the type $\mathsf{qinv}(f)$ is poorly behaved. For instance, for a single function $f : A \to B$ there may be multiple unequal inhabitants of it.

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

Theorem

$\mathsf{isequiv}(f)$ *has the following properties:*

1. *For each* $f : A \to B$ *there is a function* $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$.
2. *Similarly, for each* f *we have* $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$*; thus the two are logically equivalent.*
3. *For any two inhabitants* $e_1, e_2 : \mathsf{isequiv}(f)$ *we have* $e_1 = e_2$.

Equivalence (cont.)

We write $(A \simeq B)$ for the type of equivalences from A to B, i.e. the type

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

Theorem

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

1. *For any A, the identity function id_A is an equivalence; hence $A \simeq A$.*
2. *For any $f : A \simeq B$, we have an equivalence $f^{-1} : B \simeq A$.*
3. *For any $f : A \simeq B$ and $g : B \simeq C$, we have $g \circ f : A \simeq C$.*

The higher groupoid structure of type formers

In chapter 1, we introduced many ways to form new types: cartesian products, disjoint unions, dependent products, dependent sums, etc.
In sections 2.1-2.3, we saw that *all* types in homotopy type theory behave like spaces or higher groupoids.
Our goal in the rest of the chapter is to make explicit how this higher structure behaves in the case of the particular types defined in chapter 1.

Equality types

It turns out that for many types A, the equality types $x =_A y$ can be characterized, up to equivalence, in terms of whatever data was used to construct A. For example, if A is a cartesian product $B \times C$, and $x \equiv (b, c)$ and $y \equiv (b', c')$, then we have an equivalence

$$\big((b, c) = (b', c')\big) \simeq \big((b = b') \times (c = c')\big).$$

Transport

Similarly, when a type family $P : A \to \mathcal{U}$ is built up fiberwise using the type forming rules from chapter 1, the operation $\mathsf{transport}^P(p, -)$ can be characterized, up to homotopy, in terms of the corresponding operations on the data that went into P. For instance, if $P(x) \equiv B(x) \times C(x)$, then we have

$$\mathsf{transport}^P(p, (b, c)) = \big(\mathsf{transport}^B(p, b), \mathsf{transport}^C(p, c)\big).$$

Functoriality

Finally, the type forming rules are also functorial, and if a function f is built from this functoriality, then the operations ap_f and apd_f can be computed based on the corresponding ones on the data going into f.
For instance, if $g : B \to B'$ and $h : C \to C'$ and we define $f : B \times C \to B' \times C'$ by $f(b,c) :\equiv (g(b),h(c))$, then modulo the equivalence $\big((b,c) = (b',c')\big) \simeq \big((b = b') \times (c = c')\big)$, we can identify ap_f with "$(\mathsf{ap}_g, \mathsf{ap}_h)$".

Remarks

It would seem to be more convenient and intuitive if these characterizations of identity types, transport, and so on were *judgmental* equalities.
However, in the theory presented in chapter 1 the identity types are defined uniformly for all types by their induction principle, so we cannot “redefine” them to be different things at different types.
Thus, the characterizations for particular types to be discussed in this chapter are, for the most part, *theorems* which we have to discover and prove, if possible.

Remarks (cont.)

Actually, the type theory of chapter 1 is insufficient to prove the desired theorems for two of the type formers: Π-types and universes.
We are forced to introduce axioms into our type theory in order to make those “theorems” true.
The axiom for Π-types is familiar to type theorists: it is called *function extensionality*, and states (roughly) that if two functions are homotopic in the sense of section 2.4, then they are equal.
The axiom for universes, however, is a new contribution of homotopy type theory due to Voevodsky: it is called the *univalence axiom*, and states (roughly) that if two types are equivalent in the sense of section 2.4 then they are equal.

Remarks (cont.)

It is important to note that not *all* identity types can be “determined” by induction over the construction of types. Counterexamples include most nontrivial higher inductive types, e.g. the spheres $\mathbb{S}^n$ (chapter 6).

Cartesian product types: identity types

Given types A and B, consider the cartesian product type $A \times B$. For any elements $x, y : A \times B$ and a path $p : x =_{A \times B} y$, by functoriality we can extract paths $\mathsf{pr}_1(p) : \mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)$ and $\mathsf{pr}_2(p) : \mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)$. Thus, we have a function

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

Then for any x and y, this function is an equivalence.
We denote its inverse constructed in the proof by

$$\mathsf{pair}^{=} : (\mathsf{pr}_1(x) = \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) = \mathsf{pr}_2(y)) \to (x = y).$$

Cartesian product types: identity types (cont.)

We can view $\mathsf{pair}^{=}$ as an *introduction rule* for $x = y$ and the two components $\mathsf{ap}_{\mathsf{pr}_1}$ and $\mathsf{ap}_{\mathsf{pr}_2}$ as *elimination* rules.
Similarly, the two homotopies which witness

$$(\mathsf{pr}_1(x) = \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) = \mathsf{pr}_2(y)) \simeq (x = y)$$

can be seen as *propositional computation rules*:

$$\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{pair}^{=}(p, q)) = p$$
$$\mathsf{ap}_{\mathsf{pr}_2}(\mathsf{pair}^{=}(p, q)) = q$$

and a *propositional uniqueness principle*:

$$r = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(r), \mathsf{ap}_{\mathsf{pr}_2}(r)) \qquad \text{for } r : x =_{A \times B} y.$$

Cartesian product types: transport

We now consider transport in a pointwise product of type families. Given type families $A, B : Z \to \mathcal{U}$, we abusively write $A \times B : Z \to \mathcal{U}$ for the type family defined by $(A \times B)(z) :\equiv A(z) \times B(z)$.
Now given $p : z =_Z w$ and $x : A(z) \times B(z)$, we can transport x along p to obtain an element of $A(w) \times B(w)$.

Theorem

In the above situation, we have

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

Cartesian product types: functoriality

Finally, we consider the functoriality of ap under cartesian products. Suppose given types A, B, A', B' and functions $g : A \to A'$ and $h : B \to B'$; then we can define a function $f : A \times B \to A' \times B'$ by $f(x) :\equiv (g(\mathsf{pr}_1 x), h(\mathsf{pr}_2 x))$.

Theorem

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

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

Σ-types: identity types

The Σ-type, or dependent pair type, $\sum_{(x:A)} P(x)$ is a generalization of the cartesian product type.
Thus, we expect its higher groupoid structure to also be a generalization of the previous section.
In particular, its paths should be pairs of paths, but it takes a little thought to give the correct types of these paths.

Theorem

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

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

Again we denote the backward direction by $\mathsf{pair}^=$.

Σ-types: transport

Theorem

Suppose we have type families

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

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

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

Σ-types: functoriality

Suppose given types A, A' and type families $B : A \to \mathcal{U}, B' : A' \to \mathcal{U}$ and functions

$$g : A \to A' \qquad \text{and} \qquad h : \prod_{x:A} B(x) \to B'(g(x));$$

then we can define a function $f : (\sum_{(t:A)} B(t)) \to (\sum_{(t:A')} B'(t))$ by $f(x) :\equiv (g(\mathsf{pr}_1 x), h_{\mathsf{pr}_1 x}(\mathsf{pr}_2 x))$.

Theorem

In the above situation, given $x, y : \sum_{(t:A)} B(t)$ *and* $p : \mathsf{pr}_1 x = \mathsf{pr}_1 y$ *and* $q : p_*(\mathsf{pr}_2 x) = \mathsf{pr}_2 y$*, we have*

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

The unit type

Theorem

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

The transport lemma for $\mathbf{1}$ is simply the transport lemma for constant type families:

Theorem

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

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