A pre-introduction to homotopy type theory

Joshua Chen

Graduate Seminar in Logic, Universität Bonn, Summer 2017

Abstract

This was a set of notes prepared for the graduate seminar on type theory at the University of Bonn in the summer of 2017. It is a *pre*-introduction to homotopy type theory (HoTT) in that everything discussed here is already present in "standard" Martin-Löf type theory; however our viewpoint is towards a more homotopical interpretation of the theory. Most of the material loosely follows the presentation given in Chapters 1 and 2 of the Homotopy Type Theory (HoTT) book.

1 Preliminaries

1.1 Type universes

I will often write things like $A : \mathcal{U}$ ("A is a type") or $B \colon A \to \mathcal{U}$ ("B is a dependent type/type family"). Here $\mathcal{U}$ denotes a type whose objects are themselves types—but to avoid Girard's paradox, *$\mathcal{U}$ only contains those types that we need, and not all types.*

More formally one defines a hierarchy of **type universes**

$$\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$$

such that $\mathcal{U}_i : \mathcal{U}_{i+1}$ and $A : \mathcal{U}_i \implies A : \mathcal{U}_{i+1}$. We may then pick a suitable level containing all the types we want to work with, and call this level $\mathcal{U}$.

1.2 Dependent types

Definition 1.1. *A* ***dependent type*** *aka* ***type family*** *is a function $B \colon A \to \mathcal{U}$ that depends on objects of some other type. That is, $B(a) : \mathcal{U}$ for all $a : A$.*

Examples:

- $\mathrm{Fin} \colon \mathbb{N} \to \mathcal{U}$, where $\mathrm{Fin}(n)$ is the finite type with n objects $0_n, 1_n, \ldots, (n-1)_n$.
- The **constant type family** at a type B

$$\lambda(x : A).B \colon A \to \mathcal{U}$$

2 Π-types

The **Π-type** aka **dependent function** or **dependent product type** is a generalization of the function type $A \to B$, where the type of the returned value can vary depending on the argument.

Its governing rules are:

Formation. If $A : \mathcal{U}$ and $B : A \to \mathcal{U}$ then we can form the type

$$\prod_{x:A} B(x) : \mathcal{U}$$

(to be read as "take argument $x : A$ and return object of type $B(x)$".)

Introduction. Let $A : \mathcal{U}$ and $B : A \to \mathcal{U}$. If assuming a variable $x : A$ we can obtain $b : B(x)$ where x is potentially free in b, then

$$\lambda(x : A).b : \prod_{x:A} B(x).$$

I will often just provide an expression b involving x and write $f(x) :\equiv b$.

Dependent functions are used in the obvious way:

Elimination. If $f : \prod_{x:A} B(x)$ and $a : A$ then $fa : B(a)$.

Computation. $(\lambda(x : A).b)a \equiv b[a/x]$ (β-reduction).

Examples:

- $f : \prod_{n:\mathbb{N}} \mathrm{Fin}(n+1)$ where $f(n) :\equiv 0_{n+1} : \mathrm{Fin}(n+1)$.
- **Polymorphic functions** are dependent functions that take types as some of their arguments, and act on objects of those types (or types constructed from those types).

  e.g. The polymorphic identity function

  $$\mathrm{id} : \prod_{A:\mathcal{U}} (A \to A)$$

  defined as $\mathrm{id} :\equiv \lambda(A : \mathcal{U}).\lambda(x : A).x$.

  e.g.

  $$\mathrm{swap} : \prod_{A:\mathcal{U}} \prod_{B:\mathcal{U}} \prod_{C:\mathcal{U}} ((A \to B \to C) \to (B \to A \to C))$$

  switches the arguments of a two-argument function:

  $$\mathrm{swap} :\equiv \lambda(A : \mathcal{U}).\lambda(B : \mathcal{U}).\lambda(C : \mathcal{U}).\lambda(f : A \to B \to C).\lambda(b : B).\lambda(a : A).f(a)(b).$$

Note that if B is a constant type family then $\prod_{x:A} B(x) \equiv A \to B$.

3 Σ-types

The **Σ-type** aka **dependent pair** or **dependent sum type** generalizes the pair type—the type of the second component can depend on the first component.

Formation. If $A : \mathcal{U}$ and $B \colon A \to \mathcal{U}$ then

$$\sum_{x:A} B(x) : \mathcal{U}$$

is a type.

Introduction. If $a : A$ and $b : B(a)$ then $(a, b) : \sum_{x:A} B(x)$.

We present the so-called "positive" form of the elimination and computation rules, which has the following statement:

Elimination & computation. Let

$$C \colon \Big(\sum_{x:A} B(x)\Big) \to \mathcal{U}$$

be a type dependent on the Σ-type. Given

$$g : \prod_{x:A} \prod_{y:B(x)} C((x, y)),$$

there is a function

$$f : \prod_{p:\sum_{x:A} B(x)} C(p)$$

satisfying $f((x, y)) \equiv g(x)(y)$.

This expresses an **induction principle**: to prove that a predicate C holds for all objects p of a Σ-type, it suffices to show that C holds for all objects (a, b) given by the constructor (introduction rule).

Stated from another viewpoint, to define a dependent function f on a Σ-type it suffices to define f on the objects (a, b). This is analogous to the case of $\mathbb{N}$, where to define a function f on $\mathbb{N}$ it suffices to define f on the constructors 0 and $\mathrm{succ}(n)$ for $n : \mathbb{N}$. We'll see induction again especially when we talk about the equality type.

From the induction principle we can show that *all* $p : \sum_{x:A} B(x)$ are of the form (a, b). We can also derive the more familiar "negative" form of the elimination rules, which say that given $p : \sum_{x:A} B(x)$ we can obtain their first and second components $\pi_1(p) : A$ and $\pi_2(p) : B(\pi_1(p))$. Define

$$\pi_1 \colon \Big(\sum_{x:A} B(x)\Big) \to A$$

by

$$g : \prod_{x:A} \prod_{y:B(x)} A$$

where $g :\equiv \lambda(x : A).\lambda(y : B(x)).x$. The case for π_2 is analogous.

Note that if B is a constant type family then $\sum_{x:A} B(x) \equiv A \times B$.

4 Semantic interpretation of Π/Σ-types

The expression

$$f : \prod_{x:A} B(x)$$

has an interpretation in (intuitionistic) predicate logic: for every $a : A$ it gives an object $fa : B(a)$, i.e. it tells us that $B(a)$ is inhabited. Hence Π corresponds to the $\forall$-quantifier: *for all* $x : A$*,* $B(x)$ *is provable.*

Similarly every

$$p : \sum_{x:A} B(x)$$

is of the form (a, b) where $a : A$ and $b : B(a)$, hence the existence of such p tells us that *there exists* $a : A$ *for which* $B(a)$ *is provable.* This corresponds to the $\exists$-quantifier.

We can also think of $\sum_{x:A} B(x)$ as the type of objects $x : A$ for which property B holds.

5 Identity types

The **equality** aka **identity type** is governed by the following rules.

Formation. Given $A : \mathcal{U}$ and $a, b : A$ we may form the type $(a =_A b) : \mathcal{U}$.

Introduction. If $a : A$ then $\mathrm{refl}_a : a =_A a$ is the **reflexive identity** for a.

The elimination-computation rule is known as **path induction**, due to the homotopy type theory viewpoint of equalities as paths (to be elaborated on later).

Path induction. Let

$$C : \prod_{x,y:A} (x =_A y \to \mathcal{U}).$$

Given

$$c : \prod_{x:A} C(x, x, \mathrm{refl}_x),$$

there is a function

$$J_{C,c} : \prod_{x,y:A} \prod_{p:x=_A y} C(x, y, p)$$

satisfying $J_{C,c}(x, x, \mathrm{refl}_x) \equiv c(x)$.

It is perhaps helpful to compare the above statement with the following elimination rule seen in a previous talk (refer Section 4.10, *Type Theory & Functional Programming*, Simon Thompson)—given $x, y : A$ we have the derivation rule

$$\frac{p : x =_A y \qquad c(x) : C(x, x, \mathrm{refl}_x)}{J_{C,c}(x, y, p) : C(x, y, p)}$$

Path induction says that to prove that $C(x, y, p)$ is inhabited for any $p : x =_A y$ it suffices to prove it for the case where $y \equiv x$ and p is $\mathrm{refl}_x : x =_A x$.

Lemma 5.1 (Equality is symmetric, aka *paths can be reversed*)**.** *Let $A : \mathcal{U}$ and $x, y : A$. There is a function*

$$\cdot^{-1} : (x =_A y) \to (y =_A x)$$

such that $\mathrm{refl}_x^{-1} \equiv \mathrm{refl}_x$ *for all* $x : A$.

Proof. We show that

$$\prod_{x,y:A} ((x =_A y) \to (y =_A x))$$

is inhabited by a function with the required property. Let $C : \prod_{x,y:A}(x =_A y \to \mathcal{U})$ be defined by

$$C(x, y, p) :\equiv (y =_A x),$$

and let

$$c :\equiv \lambda(x : A).\mathrm{refl}_x : \prod_{x:A} C(x, x, \mathrm{refl}_x).$$

By path induction we have

$$J_{C,c} : \prod_{x,y:A} \prod_{p:x=_A y} C(x, y, p) \equiv \prod_{x,y:A} ((x =_A y) \to (y =_A x)).$$

For given $x, y : A$ define

$$\cdot^{-1} :\equiv J_{C,c}(x, y),$$

then $\mathrm{refl}_x^{-1} \equiv J_{C,c}(x, x, \mathrm{refl}_x) \equiv c(x) \equiv \mathrm{refl}_x$. □

Lemma 5.2 (Equality is transitive, aka *paths can be composed*)**.** *Let $A : \mathcal{U}$ and $x, y : A$. There is a function*

$$- \centerdot - : (x =_A y) \to (y =_A z) \to (x =_A z)$$

such that $\mathrm{refl}_x \centerdot \mathrm{refl}_x \equiv \mathrm{refl}_x$ *for all* $x : A$.

Note that we concatenate paths from left to right.

Proof. For every $p : x =_A y$ we want a function of type

$$C(x, y, p) :\equiv \prod_{z:A} \prod_{q:y=_A z} (x =_A z).$$

By induction it suffices to assume $y \equiv x$ and $p \equiv \mathrm{refl}_x$, and show that there is a function

$$c : \prod_{x:A} C(x, x, \mathrm{refl}_x) \equiv \prod_{x:A} \prod_{z:A} \prod_{q:x=_A z} (x =_A z).$$

We might think to take c to be the identity function on $x =_A z$, but we'll do something else. $(*)$

Let

$$E : \prod_{x,z:A} (x =_A z \to \mathcal{U})$$

be given by $E(x,z,q) :\equiv x =_A z$. Then $E(x,x,\mathrm{refl}_x) \equiv x =_A x$, and we have

$$e : \prod_{x:A} E(x,x,\mathrm{refl}_x)$$

defined by $e(x) :\equiv \mathrm{refl}_x$. By induction on $q : x =_A z$ we have

$$c :\equiv J_{E,e} : \prod_{x:A}\prod_{z:A}\prod_{q:x=_A z}(x =_A z)$$

as required, and thus also

$$J_{C,c} : \prod_{x,y:A}\prod_{p:x=_A y}\prod_{z:A}\prod_{q:y=_A z}(x =_A z).$$

Note that this last type is just

$$\prod_{x,y:A}\left((x =_A y) \to \prod_{z:A}\big((y =_A z) \to (x =_A z)\big)\right).$$

We can check that the function thus defined satisfies $\mathrm{refl}_x \centerdot \mathrm{refl}_x \equiv \mathrm{refl}_x$. □

Remark. *In the proof above we used a double induction on both $p : x =_A y$ and $q : y =_A z$ to prove the existence of a function with the property $\mathrm{refl}_x \centerdot \mathrm{refl}_x \equiv \mathrm{refl}_x$. As observed at (∗) we could have simply used induction on p, but this would instead give us a function satisfying $\mathrm{refl}_y \centerdot q \equiv q$ for all $q : y =_A z$. Similarly using induction only on q would have yielded a function satisfying $p \centerdot \mathrm{refl}_y \equiv p$ for all $p : x =_A y$.*

Path reversal and concatenation behave as expected:

Lemma 5.3. *Let $A : \mathcal{U}$, $x, y, z, w : A$ and $p : x =_A y, q : y =_A z, r : z =_A w$. Then*

i) $p = \mathrm{refl}_x \centerdot p$ *and* $p = p \centerdot \mathrm{refl}_y$.

ii) $p \centerdot p^{-1} = \mathrm{refl}_x$ *and* $p^{-1} \centerdot p = \mathrm{refl}_y$.

iii) $(p^{-1})^{-1} = p$.

iv) $p \centerdot (q \centerdot r) = (p \centerdot q) \centerdot r$.

Proofs omitted, again they all use path induction.

It is important to note that the lemma above gives us **equalities (=) between equality objects within the type theory**, as opposed to definitional equivalences (≡) on the level of the metatheory.

6 Type theory and homotopy theory

Here we make more explicit the connection between type theory and homotopy theory hinted at in the previous section. The basic idea is:

Type theory	Homotopy theory
$A : \mathcal{U}$	A is a topological space.
$a : A$	$a \in A$ is a point in A.
$p : a =_A b$	p is a path between a and b in A.

But it goes deeper. In topology, paths between points a, b can have (endpoint-preserving) homotopies between them. Homotopies are simply higher-dimensional paths, so we can form homotopies between homotopies, homotopies between homotopies between homotopies. . .

In type theory, identities $p, q : a =_A b$ can potentially themselves be identified, forming higher identities $\mathfrak{p} : p =_{a=_A b} q$, $\mathscr{P} : \mathfrak{p} =_{p=q} \mathfrak{q}$, etc.

The structure in both settings is that of a **weak ∞-groupoid**—a category having morphisms between morphisms (2-*morphisms*), morphisms between morphisms between morphisms (3-*morphisms*). . . , in general, $(k+1)$-morphisms between k-morphisms for all $k \in \mathbb{N}$. These satisfy certain laws, e.g. at every level k the k-morphisms satisfy invertibility, left and right unit laws, associativity etc., **up to $(k+1)$-morphisms**.

Comparing the statement of Lemma 5.3 with the following basic result from homotopy theory helps make some of this equivalent structure clear:

Lemma 6.1 (Lemma 5.3, topological translation)**.** *Let A be a topological space, $x, y, z, w \in A$ and p, q, r paths from x to y, y to z and z to w respectively. Then*

i) $p \sim \mathrm{id}_x \centerdot p$ *and* $p \sim p \centerdot \mathrm{id}_y$.

ii) $p \centerdot p^{-1} \sim \mathrm{id}_x$ *and* $p^{-1} \centerdot p \sim \mathrm{id}_y$.

iii) $(p^{-1})^{-1} \sim p$.

iv) $p \centerdot (q \centerdot r) \sim (p \centerdot q) \centerdot r$.

where $\sim$ means "is homotopic to", id_x is the constant path at x and p^{-1} is the inverse path to p.

In both versions of the lemma, the identifications are all up to higher-level morphisms—equalities between equalities in the type theory version, and homotopies in the topological version. This explains the HoTT convention of calling identity objects "paths".

Lemma 6.2 (Functions respect equality, aka they *preserve paths*)**.** *Let $A, B : \mathcal{U}$, $f : A \to B$ and $x, y : A$. There is a function*

$$\mathrm{ap}_f : (x =_A y) \to (fx =_B fy)$$

satisfying $\mathrm{ap}_f(\mathrm{refl}_x) \equiv \mathrm{refl}_{fx}$ *for all* $x : A$.

Proof. Let $C(x, y, p) :\equiv (fx =_B fy)$. As usual it suffices to assume $y \equiv x$ and $p \equiv \mathrm{refl}_x$, and exhibit

$$c : \prod_{x:A} (fx =_B fx).$$

But $c(x) :\equiv \mathrm{refl}_{fx}$ is such a function, and the result follows by induction. $\square$

It is instructive to consider the above lemma topologically. We call ap_f the *application of f to the path.*

There is much more to say here about connections to homotopy theory, particularly with regard to the notion of *transport* and the topological interpretation of type families as fibrations (refer Section 2.3 of the HoTT Book).

7 Homotopies and equivalences

In this section we consider notions of "equality"—other than the identity type—for functions and types.

Definition 7.1. *Let $P\colon A \to \mathcal{U}$ and $f, g : \prod_{x:A} P(x)$. A **homotopy** from f to g is a dependent function of the type*

$$f \sim g :\equiv \prod_{x:A} (fx = gx).$$

Motivation: two functions f, g should be considered "equal" if their values agree on their domain.

Note that this is different from saying $f = g$. Using path induction one can show that

$$(f = g) \to (f \sim g)$$

is inhabited. With the univalence axiom (defined later) we can obtain the reverse implication, which will make the types $f = g$ and $f \sim g$ equivalent.

Lemma 7.2. *Homotopy is an equivalence relation on each dependent function type. That is, for $A : \mathcal{U}$, $P\colon A \to \mathcal{U}$ the following types are inhabited:*

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

(Proof omitted.)

We might wish to call two types A, B "equal" if there are functions $f\colon A \to B$ and $g\colon B \to A$ such that their compositions are pointwise equal to the identity, i.e. if $f \circ g \sim \mathrm{id}_B$ and $g \circ f \sim \mathrm{id}_A$.

Definition 7.3. *Let $A, B : \mathcal{U}$ and $f\colon A \to B$. A **quasi-inverse** of f is an inhabitant of the type*

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

That is, a quasi-inverse of f is a triple (g, H, K) consisting of $g\colon B \to A$ and homotopies $H\colon f \circ g \sim \mathrm{id}_B$ and $K\colon g \circ f \sim \mathrm{id}_A$.

Topologically one would expect to call such f, g "homotopy equivalences". However as quasi-inverses alone do not suffice to define univalence in a consistent way, we instead make the following definition.

Definition 7.4. *Let $A, B : \mathcal{U}$ and $f\colon A \to B$. Define the type*

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

In words, f *is an equivalence* if it has right and left homotopy inverses g, h.

In homotopy theory, given such a pair g, h one can show that g (resp. h) is also a left (resp. right) homotopy inverse, i.e. the existence of *a priori* distinct left and right inverses implies the existence of a two-sided inverse. In HoTT we have the analogous result:

Lemma 7.5. *For every $f\colon A \to B$ there is a function* $\mathrm{qinv}(f) \to \mathrm{isequiv}(f)$ *and a function* $\mathrm{isequiv}(f) \to \mathrm{qinv}(f)$*.*

Proof. Clearly the function sending a quasi-inverse (g, H, K) to (g, H, g, K) is an inhabitant of $\mathrm{qinv}(f) \to \mathrm{isequiv}(f)$.

Suppose $(g, H, h, K)\colon \mathrm{isequiv}(f)$. That is, we have $g, h\colon B \to A$, $H\colon f \circ g \sim \mathrm{id}_B$ and $K\colon h \circ f \sim \mathrm{id}_A$. Let $\gamma\colon g \sim h$ be the homotopy given by the path composition

$$g \equiv \mathrm{id}_A \circ g \overset{K^{-1} \circ g}{\sim} h \circ f \circ g \overset{h \circ H}{\sim} h \circ \mathrm{id}_B \equiv h,$$

i.e.

$$\gamma(x) :\equiv (K^{-1} g x) \centerdot (\mathrm{ap}_h H x)\colon gx = hx$$

where $K^{-1}\colon \mathrm{id}_A \sim h \circ f$ is the inverse homotopy to K (refer Lemma 7.2). Define $K'\colon g \circ f \sim \mathrm{id}_A$ by

$$K' :\equiv (\gamma f x) \centerdot (K x).$$

Then (g, H, K') is a quasi-inverse of f. $\square$

Definition 7.6. *Let $A, B\colon \mathcal{U}$. An* ***equivalence*** *from A to B is a function $f\colon A \to B$ together with a proof of* $\mathrm{isequiv}(f)$*. We write*

$$A \simeq B :\equiv \sum_{f\colon A \to B} \mathrm{isequiv}(f)$$

and say that A ***and*** *B* ***are equivalent types*** *if $A \simeq B$ is inhabited.*

Lemma 7.5 says that to prove $f\colon A \to B$ is an equivalence it is necessary and sufficient to show that it has a quasi-inverse.

Type equivalence is an equivalence relation on $\mathcal{U}$, that is:

Lemma 7.7. *For all $A, B, C\colon \mathcal{U}$,*

i) $A \simeq A$ via the identity function id_A*.*

ii) For any $f\colon A \simeq B$ there is an equivalence $f^{-1}\colon B \simeq A$.

iii) If $f\colon A \simeq B$ and $g\colon B \simeq C$ then $g \circ f\colon A \simeq C$.

(Proof omitted.)