Draft

Dimitrios Koutsoulis
11838639

May 14, 2019

1 Type Theory

1.1 Introduction

Type theory is a formal language and deductive system, that is self sufficient in the sense that it need not be formulated as a collection of axioms on top of some other formal system like First Order Logic, instead its deductive system can be built on top of its own formal language.

Central to Type Theory is the notion of *Type*. Every term a in Type Theory we come across, must lie in some type A, which we denote as $a : A$. Note that the relation $:$ is transitive, so $a : A$ and $A : B$ imply $a : B$.

For the deductive part of Type Theory, we interpret propositions as types. Proving proposition P amounts to providing some inhabitant $p : P$.

1.2 Type Construction Operations

Let's have a look at some important type constructions.

- Given types $A, B : \mathcal{U}$ we can define the type $A \to B$ of functions from A to B. We can use λ-abstraction to construct elements of this type. $\lambda x.\Phi$ lies in $A \to B$ iff for $a : A$ we have $\Phi[a/x] : B$. For $f : A \to B$ and $a : A$ we have that the application of f on a, denoted as $f(a)$ or $f\ a$, lies in B.
- Given some type $A : \mathcal{U}$ and a family of types B over A, $B : A \to \mathcal{U}$, we have the type of dependent functions
$$\prod_{a:A} B(a)$$
where for $f : \prod_{a:A} B(a)$ and $x : A$ we have $f(x) : B(x)$. As in the case of non-dependent functions, we can use lambda abstraction to construct elements of a dependently-typed function type.

- Given $A, B : \mathcal{U}$ we can define the product type $A \times B : \mathcal{U}$. For $a : A$ and $b : B$ we have the pair $(a, b) : A \times B$. We also have the projection functions
$$\texttt{pr}_1 : A \times B \to A : (a, b) \mapsto a$$
$$\texttt{pr}_2 : A \times B \to B : (a, b) \mapsto b$$

- Given $A : \mathcal{U}$ and family of types B over A, $B : A \to \mathcal{U}$, we can define the dependent pair type
$$\sum_{a:A} B(a)$$
Given $x : A$ and $b : B(x)$ we can construct the pair $(x, b) : \sum_{a:A} B(a)$. We have two projection functions, similar to the case of the product type.

- Given $A, B : \mathcal{U}$ we can construct the coproduct type $A+B$. We can construct elements of $A + B$ using the functions
$$\texttt{inl} : A \to A + B$$
$$\texttt{inr} : B \to A + B$$
This induces the induction principle
$$\texttt{ind}_{A+B} : \prod_{C:A+B\to U} \Big(\prod_{a:A} C(\texttt{inl}\ a)\Big) \to \Big(\prod_{b:A} C(\texttt{inr}\ b)\Big) \to \prod_{x:A+B} C(x)$$

- Given $x, y : A$ we have the **identity type** $x =_A y$. An element of this type amounts to a proof that x and y are equal. Say x and y are judgmentally equal. This is captured by the element $\texttt{idp}_x : x =_A y$. The relevant induction principle describes how we can use elements of an identity type
$$\texttt{ind}_{=_A} : \prod_{C:\prod_{x,y:A}(x=_A y)\to\mathcal{U}} \Big(\prod_{x:A} C(x, x, \texttt{idp}_x)\Big) \to \prod_{x,y:A} \prod_{(p:x=_A y)} C(x, y, p)$$
The relevant computation gives us the judgmental equality
$$\texttt{ind}_{=_A}(C, c, x, x, \texttt{idp}_\texttt{x}) \equiv c\ x$$
We can concatenate those paths whose domains and codomains allow for it. Paths are equivalences. That is if $p : x = y$ is such a path, we can provide its inverse p^{-1} for which we have in turn a path between $p \cdot p^{-1}$ and $\texttt{idp}_y$ and another one between $p^{-1} \cdot p$ and $\texttt{idp}_x$.

- For every type A there is its **propositional truncation** $\|A\|$. For every element $a : A$ there exists $|a| : \|A\|$. For every $x, y : \|A\|$ we have $x = y$. Given mere proposition B and $f : A \to B$, the recursion principle gives us $g : \|A\| \to B$ such that $g(|a|) \equiv f(a)$ for all $a : A$.

1.3 Important types

Definition 1.1. *We call a type A a **mere proposition** if for every $a, b : A$ we have $a = b$.*
Definition 1.2. *We call a type A a **set** if for every $a, b : A$ we have that $a =_A b$ is a mere proposition.*
Definition 1.3. *We call a type A **contractible** if there exists $a : A$ such that for all $x : A$ it holds that $x = a$.*

1.4 Logic

Our informal deductions in Type Theory will be reminiscent of First Order Logic ones. To be able to use a similar verbiage, we will set down a handful of types, corresponding to the connectives that let us form well-formed formulas in FOL. These types need to be mere propositions, so that we can form non-constructive deductions. This approach is called 'propositions as h-propositions' in the HoTT book. In the following, A and B are mere propositions.

- When we talk of conjunction $A \wedge B$, we mean the product $A \times B$.
- We interpret $A \vee B$, as the truncation $\|A + B\|$.
- We interpret $\forall a \in A,\ P(a)$, where $P(a)$ is a mere proposition for all $a \in A$, as $\prod_{a:A} P(a)$.
- We interpret $\exists a \in A,\ P(a)$, where $P(a)$ is a mere proposition for all $a \in A$, as $\|\Sigma_{a:A} P(a)\|$

2 Modalities

Definition 2.1. *A **modality** is a function $\bigcirc : \mathcal{U} \to \mathcal{U}$ with the following properties.*

1. *For every type A we have a function $\eta^{\bigcirc}_A : A \to \bigcirc A$*
2. *for every $A : \mathcal{U}$ and every type family $B : \bigcirc A \to \mathcal{U}$ we have a function*
$$\mathtt{ind}_{\bigcirc} : \Big(\prod_{a:A} \bigcirc(B(\eta^{\bigcirc}_A\ a))\Big) \to \prod_{z:\bigcirc A} \bigcirc(B\ z)$$
3. *For every $f : \prod_{a:A} \bigcirc(B(\eta^{\bigcirc}_A\ a))$ there is a path $\mathtt{ind}_{\bigcirc}(f)(\eta^{\bigcirc}_A\ a) = f\ a$*
4. *For all $z, z' : \bigcirc A$, the function $\eta^{\bigcirc}_{z=z'} : (z = z') \to \bigcirc(z = z')$ is an equivalence.*

A modality $\bigcirc$ induces a Σ-closed reflective subuniverse.
Definition 2.2. *Given modality $\bigcirc : \mathcal{U} \to \mathcal{U}$, a **reflective subuniverse** is a 'subset' of $\mathcal{U}$ encoded by a family of h-propositions $P : \mathcal{U} \to$ `Prop` such that the following conditions hold.*

- *For $A : \mathcal{U}$, we have $P(\bigcirc A)$.*
- *For $A : \mathcal{U}$ and B such that $P(B)$, the function*

  $$\lambda f.f \circ \eta_A^{\bigcirc} : (\bigcirc A \to B) \to (A \to B)$$

  is an equivalence.

*The subuniverse is Σ-**closed** if for X such that $P(X)$ and $Q : X \to \mathcal{U}$ such that $\prod_{x:X} P(Q(x))$, we have $P(\Sigma_{x:X} Q(x))$.*

Theorem 2.3. *Reflective subuniverses are closed under products. That is for subuniverse P and $B : A \to \mathcal{U}$ such that $\prod_{a:A} P(B(a))$, we have that $P(\prod_{a:A} B(a))$.*

Proof. For $a : A$, consider $\texttt{ev}_a : (\prod_{a:A} B(a)) \to B(a)$ defined by $\texttt{ev}_a(f) :\equiv f(x)$. Since $P(B(a))$, we have

$$(\lambda f.f \circ \eta^{\bigcirc}_{\prod_{a:A} B(a)})^{-1}(\texttt{ev}_a) : \bigcirc\big(\prod_{a:A} B(a)\big) \to B(a)$$

We can now define the retraction of $\eta^{\bigcirc}_{\prod_{a:A} B(a)}$ by pattern matching as such:
For $z : (\prod_{a:A} B(a))$ and $a : A$ we have

$$(\lambda f.f \circ \eta^{\bigcirc}_{\prod_{a:A} B(a)})^{-1}(\texttt{ev}_a)(z) : B(a)$$

□

Definition 2.4. *For $B : A \to \mathcal{U}$, we call X B-**null** if the map*

$$\lambda x.\lambda b.x : X \to (B(a) \to X)$$

is an equivalence for all $a : A$.

3 LLPO

Definition 3.1. *The Lesser Limited Principle of Omniscience, states that given binary sequence $s : \mathbb{N} \to \mathbf{2}$ and the fact that there is at most one occurence of 1 in the sequence, formally*

$$\texttt{atMost1one} : \prod_{n_1:\mathbb{N}} \prod_{n_2:\mathbb{N}} s(n_1) = 1 \to s(n_2) = 1 \to n_1 = n_2$$

we can then have by the LLPO a witness for $\texttt{p}_{\texttt{odd}} \vee \texttt{p}_{\texttt{even}}$, where $\texttt{p}_{\texttt{odd}}$ is the statement that for all odd positions n, $s(n) = 0$, formally $\texttt{p}_{\texttt{odd}} \equiv \prod_{n:\mathbb{N}} \texttt{odd}(n) \to s(n) = 0$. Similarly for $\texttt{p}_{\texttt{even}}$.

LLPO can be viewed as a weaker form of the Law of Excluded Middle.

Lemma 3.2. *The Law of Excluded Middle implies the Limited Principle of Omniscience.*

Proof. By LEM we have a witness $l_1 : \texttt{p}_{\texttt{odd}} \vee \neg\texttt{p}_{\texttt{odd}}$. Since this is a coproduct, by the relevant principle of induction, it's enough to prove LLPO from the disjuncts.

- $\texttt{p}_{\texttt{odd}} \Rightarrow$ LLPO, trivially.
- $\neg\texttt{p}$, alongside LEM, implies that there exists odd $n_e : \mathbb{N}$ such that $s(n_e) = 1$. We can now provide $\prod_{n:\mathbb{N}} \texttt{even}(n) \to s(n) = 0$. Let $n : \mathbb{N}$. By the definition of s, $s(n) = 0 \vee s(n) = 1$. We invoke the principle of induction of coproducts and prove LLPO from the disjuncts.
  - $s(n) = 0$, in this case we are done.
  - $s(n) = 1$. By `atMost1one` we have $n = n_e \Rightarrow n$ even and odd which is a contradiction $c : \bot$. Ex Falso, $\texttt{efq} : \bot \to s(n) = 0$. Then $\texttt{efq}(c) : s(n) = 0$.

□