Draft

Dimitrios Koutsoulis
11838639

October 21, 2019

Chapter 1

Type Theory

1.1 Introduction

Martin-Löf's intensional *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 $a : A$. To avoid impredicativity we assume a countable hierarchy of universes $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$ which is cummulative i.e. every universe includes all previous universes and their types. While working in type theory we usually simplify our view of this hierarchy and pick some $\mathcal{U}$, of arbitrary index that we do not specify, to use as our workspace.

For the deductive part of Type Theory, we interpret propositions as types. Proving proposition P amounts to providing some inhabitant $p : P$. This is in agreement with the BHK interpretation. Type Theory is rich in type formation rules that gives us the breadth required to do Intuitionistic Logic inside of it.

1.2 Type Construction Operations

We have the following list of types at our disposal.

- Given types $A, B : \mathcal{U}$ we can define the type $A \to B : \mathcal{U}$ 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, so $f\ a : B$.

  Functions whose type is of the form $A \to \mathcal{U}$ i.e. the codomain is $\mathcal{U}$ are called *families of types*. They are of special interest because they can be viewed as types themselves; types indexed by terms of the domain type. So if $B : A \to \mathcal{U}$ is such a family of types, then its inhabitants would be the collection of inhabitants of all $B(a)$'s for all $a : A$. This last sentence is not sanctioned by formal type theory and only serves to make the introduction of the first dependent type, the *dependent function type*, easier.

- 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** or **dependent products** (which should not be viewed as the dependent version of the product type defined later on)
$$\prod_{a:A} B(a)$$
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. That way, $\lambda x.\Phi$ lies in $\prod_{a:A} B(a)$ iff B is of the form $\lambda y.\Psi$ and for all $a : A$ we have $\Phi[a/x] : \Psi[a/y]$.

- 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
$$\mathtt{pr}_1 : A \times B \to A : (a, b) \mapsto a$$
$$\mathtt{pr}_2 : A \times B \to B : (a, b) \mapsto b$$
We sometimes use the notation $x.\mathtt{pr}_i$ to refer to $\mathtt{pr}_i\ x$. We also make use of the alias `fst` for $\mathtt{pr}_1$ and `snd` for $\mathtt{pr}_2$.

We also have the following induction principle

$$\mathtt{ind}_{A\times B} : \prod_{C:A\times B\to\mathcal{U}} \left(\prod_{a:A}\prod_{b:B} C((a,b))\right) \to \prod_{x:A\times B} C(x)$$

$$\mathtt{ind}_{A\times B}(C, f, (a,b)) :\equiv f(a)(b)$$

So given two functions $f : A \to C$ and $g : B \to C$ we can construct an $h : A \times B \to C$ such that $h((a,b)) \equiv \big(f(a), g(b)\big)$ for all $a : A,\ b : B$.

- Given $A : \mathcal{U}$ and family of types B over A, $B : A \to \mathcal{U}$, we can define the **dependent pair** type (the dependent version of the product 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.

$$\mathtt{pr}_1 : \sum_{a:A} B(a) \to A : (a,b) \mapsto a$$

$$\mathtt{pr}_2 : \prod_{x:\sum_{a:A} B(a)} B(\mathtt{pr}_1\ x) : (a,b) \mapsto b$$

Like in the case of the product type, we make use of the dot notation $x.\mathtt{pr}_i$ here too, along with the aliases `fst` and `snd`.

The induction principle is the following

$$\mathtt{ind}_{\sum_{a:A} B(a)} : \prod_{C:\sum_{a:A} B(a)\to\mathcal{U}} \Big(\prod_{a:A}\prod_{b:B(a)} C((a,b))\Big) \prod_{x:\sum_{a:A} B(a)} C(x)$$

$$\mathtt{ind}_{\sum_{a:A} B(a)}(C, f, (a,b)) :\equiv f(a)(b)$$

- Given $A, B : \mathcal{U}$ we can construct the **coproduct** type $A + B$. We can construct elements of $A + B$ using the functions

$$\mathtt{inl} : A \to A + B$$

$$\mathtt{inr} : B \to A + B$$

One can guess the induction principle

$$\mathtt{ind}_{A+B} : \prod_{C:A+B\to U} \Big(\prod_{a:A} C(\mathtt{inl}\ a)\Big) \to \Big(\prod_{b:B} C(\mathtt{inr}\ b)\Big) \to \prod_{x:A+B} C(x)$$

$$\mathtt{ind}_{A+B}(C, f_A, f_B, \mathtt{inl}\ a) :\equiv f_A\ a$$
$$\mathtt{ind}_{A+B}(C, f_A, f_B, \mathtt{inr}\ b) :\equiv f_B\ b$$

- Given $x, y : A$ we have the **identity type** $x =_A y : \mathcal{U}$ (we ommit the index A whenever it's easily deduced). An element of this type amounts to a proof that x and y are equal and we call the element a path between x and y. Say x and y are judgmentally equal, $x \equiv y$. This is captured by the element $\mathtt{idp}_x : x =_A y$. For every $x : A$ we have $\mathtt{idp}_x : x =_A x$. The relevant induction principle describes how we can use elements of an identity type

$$\mathtt{ind}_{=_A} : \prod_{C:\prod_{x,y:A}(x=_A y)\to\mathcal{U}} \Big(\prod_{x:A} C(x, x, \mathtt{idp}_x)\Big) \to \prod_{x,y:A} \prod_{(p:x=_A y)} C(x, y, p)$$

  The relevant computation rule gives us the judgmental (definitional) equality

$$\mathtt{ind}_{=_A}(C, c, x, x, \mathtt{idp_x}) \equiv c\ x$$

  We can concatenate those paths whose domains and codomains allow for it. Paths form an equivalence relation. That is, if $p : x = y$ and $p' : y = z$ are such paths then we can concatenate them $p \cdot p' : x = z$. We can also provide the inverse $p^{-1} : y = x$ for which we have in turn a path $q : (p \cdot p^{-1}) =_{x=x} (\mathtt{idp}_x)$ between $p \cdot p^{-1}$ and $\mathtt{idp}_x$ and another one between $p^{-1} \cdot p$ and $\mathtt{idp}_y$.

  This would be a good place to talk a bit more about the closure properties of judgmental/definitional equality $\equiv$. Essentially, we demand that $\equiv$ is a congruence relation built upon those judgmental equalities we introduce elsewhere, as part of enriching our type theory with type formers, in addition to being closed under these:

  - For terms $\lambda x.\ t$ and u such that the application $(\lambda x.\ t)\ u$ is legal according to their typings, we have $(\lambda x.\ t)\ u \equiv t[u/x]$.
  - If $t \equiv t'$ and $s \equiv s'$ and the application $t(s)$ is legal, then $t(s) \equiv t'(s')$.

– Closed under lambda abstraction, i.e. if $t \equiv t'$ then $\lambda x.\, t \equiv \lambda x.\, t'$.

After closing under the above, we drop from $\equiv$ all pairs of terms that belong to (judgmentally) unequal types.

We sometimes introduce judgmental equalities with a colon on the left $:\equiv$. This serves merely as an indication of 'direction' in the definition wherein the left hand side is being introduced as a term while it's well understood that the right hand side is a term of type theory. Sometimes the lhs is just a name, a shorthand, for the rhs. The colon serves no *formal* purpose and the reader may ignore it, if they do not find it helpful.

When working in type theory, we may freely replace terms with those judgmentally equal to them.

Definition 1.2.1. *We call a type A a **mere proposition** or simply a **proposition**, if for every $a, b : A$ we have $a = b$.*

- For every type $A : \mathcal{U}$ there exists its **propositional truncation** $\|A\|$. We also have the truncation map $|\cdot| : A \to \|A\|$ so that for every element $a : A$ there exists $|a| : \|A\|$. Finally, $\|A\|$ is a mere proposition.

  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 $a : A$. This recursion principle will prove itself an indispensable tool in the sections to come. Whenever, in the midst of a proof, the current goal B is a mere proposition and we have access to some witness $a : \|A\|$, we are allowed by the recursion principle to assume that we have $a' : A$ and use it to construct a proof of B.

- We have an **empty** type $\bot : \mathcal{U}$ with the induction principle

$$\mathtt{ind}_\bot : \prod_{A:\mathcal{U}} \prod_{C:A\to\mathcal{U}} \prod_{x:\bot} C\ x$$

- We have a unit type $\mathbf{1} : \mathcal{U}$ for which we have a term $* : \mathbf{1}$ and a witness of $\prod_{x,y:\mathbf{1}} x = y$.

- We have the $\mathbf{2}$ type with terms $0 : \mathbf{2}$ and $1 : \mathbf{2}$ and the induction principle

$$\mathtt{ind}_{\mathbf{2}} : \prod_{C:\mathbf{2}\to\mathcal{U}} C(0) \to C(1) \to \prod_{x:\mathbf{2}} C(x)$$

$$\mathtt{ind}_{\mathbf{2}}()$$

- We have the type of **natural numbers** $\mathbb{N} : \mathcal{U}$ equiped with $0 : \mathbb{N}$ and $\mathtt{succ} : \mathbb{N} \to \mathbb{N}$. There is also the induction principle

$$\mathtt{ind}_{\mathbb{N}} : \prod_{C:\mathbb{N}\to\mathcal{U}} C(0) \to \big(\prod_{:\mathbb{N}} C(n) \to C(\mathtt{succ}\ n)\big) \to \prod_{n:\mathbb{N}} C\ n$$

$$\mathtt{ind}_{\mathbb{N}}(C, f_0, f_s, 0) :\equiv f_0$$
$$\mathtt{ind}_{\mathbb{N}}(C, f_0, f_s, \mathtt{succ}\ n) :\equiv f_s\big(n, \mathtt{ind}_{\mathbb{N}}(C, f_0, f_s, n)\big)$$

This concludes our informal presentation of the primitives of Type Theory. We will continue with definitions and results using the tools we laid out above.

Remark 1.2.2. Sometimes, when constructing a term of some type, we say that we *take cases* on it or some other type. By this we mean that we are invoking a type's induction or recursion principle. As an example, suppose we have $A, B, A + B : \mathcal{U}$ and $C : A + B \to \mathcal{U}$ and want to construct a term of $\prod_{x:A+B} C(x)$. We can do so by taking cases on $A + B$, i.e. provide an $f_A : \prod_{a:A} C(\mathtt{inl}\ a)$ which 'describes where in C we send elements of A' and an $f_B : \prod_{b:B} C(\mathtt{inr}\ b)$. We have effectively made use of the induction principle of the coproduct.

Remark 1.2.3. We sometimes say that we can decide whether $A + B$. This is just a shorthand for 'we can construct a witness of $A + B$' and is inspired by the saying 'we can decide P' where P is a mere proposition and we mean that we can prove $P + (P \to \bot)$ which would also be a mere proposition.

Remark 1.2.4. Induction principles might appear to give us access only to dependent functions. They actually give us access to non-dependent functions too. Let $C : A \to \mathcal{U}$ be the family of types that forms the codomain of the resulting function of the induction principle. If for all $a : A$, $C(a) \equiv B$ where $B : \mathcal{U}$ then we can form a non-dependent function $f : A \to B$.

Lemma 1.2.5. *Applying any function $f : A \to B$ to $a_1, a_2 : A$ such that $p : a_1 =_A a_2$ gives us a path $\mathtt{ap}_f(p) : f(a_1) =_B f(a_2)$. We name this* ***action on paths*** *and this instance, action of f on p.*
Formally,

$$\mathtt{ap} : \prod_{A,B:\mathcal{U}} \prod_{f:A\to B} \prod_{a_1,a_2:A} (a_1 =_A a_2) \to \big(f(a_1) =_B f(a_2)\big)$$

Proof. Assume the hypotheses A, B, f, a_1, a_2 and $p : a_1 =_A a_2$ as above. We need a witness of $f(a_1) =_B f(a_2)$. We use induction on the identity type $a_1 =_A a_2$. So we have to show that for $\mathtt{idp}_{a_1} : a_1 =_A a_2$, we have $f(a_1) =_B f(a_2)$. By $\mathtt{idp}_{a_1}$ we have $a_1 \equiv a_2$ which means we can rewrite the goal as $f(a_1) =_B f(a_1)$, for which we have a witness in the form of $\mathtt{idp}_{f(a_1)}$. □

1.3 Equivalence

Before going over what it means for two types to be equivalent, we have to work out some preliminary notions.

Definition 1.3.1. *Let $f, g : \prod_{a:A} B(a)$ where $B : A \to \mathcal{U}$. We call a function of the following type a **homotopy** between f and g*

$$f \sim g :\equiv \prod_{a:A} f(a) = g(a)$$

Definition 1.3.2. *We call a type A a **set** if for every $a, b : A$ we have that $a =_A b$ is a mere proposition.*

Definition 1.3.3. *We call a type A **contractible** if there exists $a : A$ such that for all $x : A$ it holds that $x = a$.*
Formally

$$\mathtt{isContr}\ A :\equiv \sum_{x:A} \prod_{a:A} x = a$$

Definition 1.3.4. *Given some map $f : A \to B$, a **fiber** of it over some point $y : B$ is*

$$\mathtt{fib}_f\ y :\equiv \sum_{x:A} \big(f(x) = y\big)$$

Definition 1.3.5. *We say that a map $f : A \to B$ **has contractible fibers** if for every $b : B$, the type $\mathtt{fib}_f(b)$ is contractible.*
Formally,

$$\mathtt{hasContrFibers}\ f :\equiv \prod_{b:B} \mathtt{isContr}(\mathtt{fib}_f\ b)$$

One way to see this, albeit naive from a set-theoretic point of view, is that we require for every element of the codomain to have exactly one element

of the domain mapped to it by f. Note that for any map f, the type $\texttt{hasContrFibers(f)}$ is a mere proposition.

We form a notion of **equivalence of types** based on maps with contractible fibers. Whenever we have some $f : A \to B$ with contractible fibers, we say that the types A and B are equivalent and write $A \simeq B$ which we define as

$$A \simeq B :\equiv \sum_{f:A\to B} \texttt{hasContrFibers}\ f$$

To motivate this, note that whenever we have $A \simeq B$, we can form $f : A \to B$ and $g : B \to A$ so that $f \circ g \sim \texttt{id}_B$ and $g \circ f \sim \texttt{id}_A$. f and g are called each other's quasi-inverse. It also holds the other way around, if there exist f, g like above, then both of them have contractible fibers. Another way to show that $A \simeq B$ is to provide some $f : A \to B$ and $g_1, g_2 : B \to A$ such that $f \circ g_1 \sim \texttt{id}_B$ and $g_2 \circ f \sim \texttt{id}_A$. In these situations, g_1 is called a right inverse of f and g_2 a left inverse. Finally, the equivalence of types we introduced just now is an equivalence relation on $\mathcal{U}$. More exposition on equivalence can be found in chapter 4 of [5].

We will now see some Type Theory variants that expand a bit upon what we've laid out in this section. Before embarking on that, we present the notion of **function extensionality** which we do not require to hold in the type theory presented in this chapter, yet holds in both of the variants below. As a prerequisite to it, we define the function

$$\texttt{happly} : (f = g) \to (f \sim g)$$

for arbitrary functions $f, g : \prod_{a:A} B\ a$, where $A : \mathcal{U},\ B : A \to \mathcal{U}$. We do so using induction on the identity path $f = g$ which reduces it to providing a witness for $f\ a =_{B(a)} g\ a$ assuming $f \equiv g$. Clearly $f\ a \equiv g\ a$ so $\texttt{idp}_{f(a)} : f\ a =_{B(a)} g\ a$.

Axiom 1.3.6 (Function extensionality)**.** $\texttt{happly}$ *has contractible fibers.*

Function extensionality, when true, allows us to equate functions that agree on all inputs.

1.4 Univalent Type Theory

We get the flavour of Univalent Type Theory (UTT) that interests us by assuming the axiom of Univalence.
Lemma 1.4.1. *We can define the following function*

$$\texttt{idtoequiv} : (A =_{\mathcal{U}} B) \to (A \simeq B)$$

Proof. The definition of `idtoequiv` can be found in section 2.10 of [5]. □

Axiom 1.4.2 (Univalence)**.** `idtoequiv` *has contractible fibers.*

In UTT we usually assume that the universe $\mathcal{U}$ that we are working in is univalent which means that for all $A, B : \mathcal{U}$

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

Function extensionality follows from univalence.

1.5 (Definitionally) Extensional Type Theory

Extensional Type Theory (ETT) is not consistent with the Univalent Type Theory defined above. To get ETT we assume the following axiom.
Axiom 1.5.1. *Whenever we have an inhabitant $p : a =_C b$ we can infer $\texttt{idp}_a : a =_C b$ i.e. $a \equiv b$.*

This axiom simplifies the landscape considerably. Starting with it and using induction on identity types one can eventually deduce that $p \equiv \texttt{idp}_a$. Thus, the higher path structure of types collapses and types behave similarly to sets. Additionally, function extensionality follows from the above axiom.

This is enough talk about variants of Type Theory.

1.6 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. When working intuitionistically we might say that we have *actual*/there exists *actual* $a : A$ of some $A : \mathcal{U}$ and this constitutes a constructive proof of A under the 'propositions as types' regime as seen in [5]. In accordance to this and the BHK interpretation, we have the following translation of FOL connectives and quantifiers. Assume $A, B : \mathcal{U}$ and $C : A \to \mathcal{U}$.

- $A \wedge B$ is $A \times B$.
- $A \vee B$ is $A + B$.
- $\neg A$ is $A \to \bot$.
- $\forall a \in A,\ C(a)$ is simply translated to $\prod_{a:A} C(a)$.
- Finally, existence $\exists a \in A,\ C(a)$ is interpreted as $\sum_{a:A} C(a)$.

We would also like a way to do classical logic in Type Theory. To do so we would need a way to say that some type C is inhabited without providing a specific witness $c : C$, which would have the undesirable effect of providing more information than a classical statement of existence. An effective way to do so is to provide a witness of the truncation $\|C\|$ and say that C is *merely* inhabited. So to work classically we restrict ourselves to using only mere propositions. This approach is called 'propositions as mere propositions' in [5].

- When we talk of conjunction $A \wedge B$, where A and B are mere propositions, we mean the product $A \times B$.
- We interpret $A \vee B$, where A and B are mere propositions, as the truncation $\|A + B\|$.
- $\neg A$ is $A \to \bot$.
- 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 $\| \sum_{a:A} P(a) \|$

Note that the above are all chosen so that they preserve mere propositions, e.g. $A \times B$ is a mere proposition. The use of the above notation is inter-

changeable in the sections to follow. We will try to specify whether a type is *merely* or *actually* inhabited to avoid ambiguity whenever needed.

1.7 Formal Type Theory

We shall now have a look at what Intensional Type Theory *actually is*. We will give a short description of a formal system in which the informal derivations we do in the rest of this text are meant to be taking place. Building blocks of the formal system will be juxtaposed against their informal counterparts. We direct those looking for a full description to Appendixes A.1 and A.2 of [5].

When working informally, at any given point we have a collection of assumptions about variables and their typings that are available to be used in completing the next step of our derivation. This is captured by the *context* which is a list of typings $x_1 : A_1,\ x_2 : A_2,\ \ldots,\ x_n : A_n$ of variables x_i and their types A_i. Each x_i must be distinct.

Example 1.7.1. As one would guess

$$x_1 : \mathbf{2},\ x_2 : \mathbf{2},\ x_3 : (x_1 =_{\mathbf{2}} x_2)$$

is a legitimate context.

The deductive system of TT is a sort of a sequent calculus where we infer sequents called *judgments* which are of the form $\Gamma \vdash a : A$ where Γ is a context and $a : A$ the statement that a is a term of type A. Both a and A are composed exclusively by *terms over* Γ. By terms over Γ we mean terms that can appear in the right hand side of valid judgments with Γ as the left hand side, e.g. b is a term over Γ if $\Gamma \vdash b : B$ can be inferred in our system. The rules of inferrence prescribe how we are allowed to combine said terms and derive valid judgments.

The judgment $\Gamma \vdash a : A$ corresponds to the exact step of an informal derivation where we construct the witness $a : A$ using terms over Γ which we constructed in earlier steps of the same derivation. As one might expect, the deductive system is populated with inference rules that allow us to deduce valid judgments from collections of others; to illustrate

$$\frac{J_1, J_2, \ldots, J_n}{J}$$

All the constructions and rules described in 1.2 should and can be transcribed in this form.

There is an important judgment that subverts the form described above and which asserts the correctness of a context. We write it as Γ `ctx`. There are also two rules that let us build valid contexts. One affirming the empty context's validity and can be the root of a deduction tree

$$\frac{}{\cdot \texttt{ ctx}}$$

and one which lets us extend a given context Γ with *a type over it* (a term which, under the assumptions in Γ, can be inferred to be of type $\mathcal{U}_i$ for some $i : \mathbb{N}$)

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{(\Gamma,\ x : A)\ \texttt{ctx}}$$

where x must differ from all variables in Γ.

Example 1.7.2. Consider the case of constructing a function. Suppose that the current context is $\Gamma,\ x : A$ where x is a variable. If we can deduce that $b : B$ (where b, B might not be ground terms but instead contain variables like x or those in Γ) then this corresponds to a valid judgment $\Gamma,\ x : A \vdash b : B$. Formally, this is captured by the rule

$$\frac{\Gamma,\ x : A \vdash b : B}{\Gamma \vdash \lambda(x : A).b : \prod_{x:A} B}$$

Note that the function and its type $\prod_{x:A} B$ are allowed to be dependent, as we allowed b and B to be open terms containing variables, that is why there is no point in specifying $B(x)$.

Example 1.7.3. Judgmental equalities are also taken care of by this system.

Suppose, like in the previous example that we are in the situation captured by the judgment $\Gamma,\ x : A \vdash b : B$. Suppose that we also have a term a over Γ i.e. $\Gamma \vdash a : A$. Then we should be able to deduce that application of $\lambda(x : A).\ b$ on a would be judgmentally equal to $b[a/x]$ which is the result of replacing all occurences of x in b with a. Also both should be of type $B[a/x]$. These facts are captured by the following four judgments

$$\frac{\Gamma,\ x : A \vdash b : B, \qquad \Gamma \vdash a : A}{\Gamma \vdash \big(\lambda(x : A).b\big)(a) : B[a/x]}$$

$$\frac{\Gamma,\ x : A \vdash b : B, \qquad \Gamma \vdash a : A}{\Gamma \vdash b[a/x] : B[a/x]}$$

$$\frac{\Gamma,\ x : A \vdash b : B, \qquad \Gamma \vdash a : A}{\Gamma \vdash \Big(\big(\lambda(x : A).b\big)(a) \equiv b[a/x]\Big) : \Big(b[a/x] =_{B[a/x]} b[a/x]\Big)}$$

$$\frac{\Gamma,\ x : A \vdash b : B, \qquad \Gamma \vdash a : A}{\Gamma \vdash \Big(\big(\lambda(x : A).b\big)(a) \equiv b[a/x]\Big) : \Big(\big(\lambda(x : A).b\big)(a) =_{B[a/x]} \big(\lambda(x : A).b\big)(a)\Big)}$$

The formal system has enough rules to enable deducing all the properties of judgmental equality we mentioned in the definition of identity types.

Chapter 2

Modalities

Definition 2.0.1. *A **modality** is any function $\bigcirc : \mathcal{U} \to \mathcal{U}$ with the following properties.*

1. *For every type A we have a function $\eta^{\bigcirc}_A : A \to \bigcirc A$ called the modal unit.*
2. *for every $A : \mathcal{U}$ and every type family $B : \bigcirc A \to \mathcal{U}$ we have the induction principle*

$$\mathtt{ind}_{\bigcirc} : \Big(\prod_{a:A} \bigcirc(B(\eta^{\bigcirc}_A\ a))\Big) \to \prod_{z:\bigcirc A} \bigcirc(B\ z)$$

   where A and B are implicit arguments of $\mathtt{ind}_{\bigcirc}$ and can be derived from context.

3. *For every $f : \prod_{a:A} \bigcirc(B(\eta^{\bigcirc}_A\ a))$ and every $a : A$, 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.*

Lemma 2.0.2. *Given $A : \mathcal{U}$, if $\eta^{\bigcirc}_A : A \to \bigcirc A$ has a left inverse, then $A \simeq \bigcirc A$.*

Proof. Assume the hypotheses of the lemma. We need to show that $A \simeq \bigcirc A$. We already have a left inverse so producing a right inverse for $\eta^{\bigcirc}_A$ would be enough to show equivalence. Let $f : \bigcirc A \to A$ be the left inverse, i.e.

$f \circ \eta^{\bigcirc}_A \sim \mathtt{id}_A$. We then have $\eta^{\bigcirc}_A \circ f \circ \eta^{\bigcirc}_A \sim \eta^{\bigcirc}_A \circ \mathtt{id}_A$ and $\eta^{\bigcirc}_A \circ f \circ \eta^{\bigcirc}_A \sim \mathtt{id}_{\bigcirc A} \circ \eta^{\bigcirc}_A$. This translates to

$$h : \prod_{a:A} \eta^{\bigcirc}_A \circ f\ (\eta^{\bigcirc}_A\ a) = \mathtt{id}_{\bigcirc A}\ (\eta^{\bigcirc}_A\ a)$$

We define the following function by assuming $a : A$ and applying the relevant modal unit to $h\ a$

$$h' : \prod_{a:A} \bigcirc\big(\eta^{\bigcirc}_A \circ f\ (\eta^{\bigcirc}_A\ a) = \mathtt{id}_{\bigcirc A}\ (\eta^{\bigcirc}_A\ a)\big)$$

We then use the induction principle of the modality to get

$$\mathtt{ind}_{\bigcirc}\ h' : \prod_{z:\bigcirc A} \bigcirc\big(\eta^{\bigcirc}_A \circ f\ (z) = \mathtt{id}_{\bigcirc A}\ (z)\big)$$

Then, by the equivalence mentioned in the fourth datum of 2.0.1 we have a quasi-inverse r for the modal unit $\eta_{\eta^{\bigcirc}_A \circ f\ (z) = \mathtt{id}_{\bigcirc A}\ (z)}$, which we use to construct

$$\lambda\ (z : \bigcirc A).\ r\big((\mathtt{ind}_{\bigcirc}\ h')\ (z)\big) : \prod_{z:\bigcirc A} \eta^{\bigcirc}_A \circ f\ (z) = \mathtt{id}_{\bigcirc A}\ (z)$$

We've proven $\eta^{\bigcirc}_A \circ f \sim \mathtt{id}_{\bigcirc A}$ which means that f is the right inverse of $\eta^{\bigcirc}_A$. Since the modal unit has both a left and a right inverse, we can conclude that $A \simeq \bigcirc A$. □

The usefulness of this lemma lies in that it makes it easier for us to provide a proof of equivalence between a type and its image under $\bigcirc$, when they are so. These types are of special interest to us because they form a $\sum$-closed reflective subuniverse of $\mathcal{U}$. We shall define a predicate to tell them apart.

Definition 2.0.3. *We define* $\mathtt{isModal}_{\bigcirc} : \mathcal{U} \to \mathtt{Prop}$ *as such*

$$\mathtt{isModal}_{\bigcirc} \equiv \prod_{A:\mathcal{U}} (A \simeq \bigcirc A)$$

Definition 2.0.4. *Given modality* $\bigcirc : \mathcal{U} \to \mathcal{U}$*, the* $\sum$*-closed* ***reflective subuniverse*** *of* $\mathcal{U}$ *is encoded by the following type*

$$\mathcal{U}^{\bigcirc} \equiv \sum_{A:\mathcal{U}} \mathtt{isModal}_{\bigcirc}(A)$$

That the subuniverse is $\sum$***-closed*** *means that for* X *such that* $\mathtt{isModal}_{\bigcirc}(X)$ *and* $Q : X \to \mathcal{U}$ *such that* $\prod_{x:X} \mathtt{isModal}_{\bigcirc}(Q(x))$*, we have* $\mathtt{isModal}_{\bigcirc}(\Sigma_{x:X} Q(x))$.

The *reflective* modifier refers to the fact that for each $f : A \to B$ there is a canonical way to construct its *reflection*, $f' : \bigcirc A \to \bigcirc B$. We simply compose f with the modal unit $\eta^{\bigcirc}_B$ and use the induction principle of the modality on the result.

Propositional truncation is a modality as it possesses the required data outlined in 2.0.1. Its reflective subuniverse is the universe of mere propositions.

Proposition 2.0.5. *For all* $A : \mathcal{U}$*, we have* $\mathtt{isModal}_{\bigcirc}(\bigcirc A)$.

Lemma 2.0.6. *For* $A, B : \mathcal{U}$ *such that* $\mathtt{isModal}\ B$ *we have*

$$(A \to B) \simeq (\bigcirc A \to B)$$

Proof. Note that $(A \to B) \simeq (A \to \bigcirc B)$ and $(\bigcirc A \to B) \simeq (\bigcirc A \to \bigcirc B)$, since B is modal. So we reduce our goal to

$$(A \to \bigcirc B) \simeq (\bigcirc A \to \bigcirc B)$$

We need only provide a quasi-inverse for

$$(- \circ \eta^{\bigcirc}_A) : (\bigcirc A \to \bigcirc B) \to (A \to \bigcirc B)$$

to conclude the proof. We propose $\mathtt{ind}$ as the quasi-inverse. We first show $\left(\mathtt{ind} \circ (- \circ \eta^{\bigcirc}_A)\right) \sim \mathtt{id}_{\bigcirc A \to \bigcirc B}$. Let $g : \bigcirc A \to \bigcirc B$. We need $\mathtt{ind}(g \circ \eta^{\bigcirc}_A) = g$. By function extensionality it's enough to show that

$$\prod_{x:\bigcirc A} \mathtt{ind}(g \circ \eta^{\bigcirc}_A)(x) = g(x)$$

So, for every x, we are trying to provide a witness for an identity path of terms in $\bigcirc B$. By the fourth datum of 2.0.1 and the relevant induction principle, with implicit arguments A and $x \mapsto \mathtt{ind}(g \circ \eta^{\bigcirc}_A)(x) = g(x) : \bigcirc A \to \mathcal{U}$, we can reduce it to constructing a witness for

$$\prod_{a:A} \mathtt{ind}(g \circ \eta^{\bigcirc}_A)(\eta^{\bigcirc}_A\ a) = g(\eta^{\bigcirc}_A\ a)$$

A witness of this type is secured for us, once again by the definition of modalities, the third item with $g \circ \eta^{\bigcirc}_A$ as f.

The proof of $\big((- \circ \eta^{\bigcirc}_A) \circ \mathtt{ind}\big) \sim \mathtt{id}_{A \to \bigcirc B}$ is similar to the above, with some steps being even more immediate. □

It is easy to generalize the above lemma to dependent functions.

Corollary 2.0.7. *For $A : \mathcal{U}$ and $B : \bigcirc A \to \mathcal{U}$ such that $\prod_{x:\bigcirc A} \mathtt{isModal}_{\bigcirc}(B\ x)$ we have*

$$\Big(\prod_{a:A} B(\eta^{\bigcirc}_A\ a)\Big) \simeq \Big(\prod_{x:\bigcirc A} B(x)\Big)$$

Theorem 2.0.8. *Reflective subuniverses are closed under dependent products. That is, for the subuniverse of $\bigcirc$ and $B : A \to \mathcal{U}$ such that $\prod_{a:A} \mathtt{isModal}_{\bigcirc}(B(a))$, we have that $\mathtt{isModal}_{\bigcirc}(\prod_{a:A} B(a))$.*

Proof. By 2.0.2 it is enough to provide a left inverse for

$$\eta^{\bigcirc}_{\prod_{a:A} B(a)} : \Big(\prod_{a:A} B(a)\Big) \to \bigcirc\Big(\prod_{a:A} B(a)\Big)$$

First, for $a : A$, consider $\mathtt{ev}_a : (\prod_{a:A} B(a)) \to B(a)$ defined by $\mathtt{ev}_a(f) :\equiv f(a)$. By $\mathtt{isModal}\ B(a)$ we have that there exists $\eta^{-1}_{B\ a}$, quasi-inverse of $\eta_{B\ a}$. We define

$$h :\equiv \lambda(f : \prod_{a:A} B\ a).\ \lambda(a : A).\ \eta^{-1}_{B\ a} \circ (\mathtt{ind}_{\prod_{x:A} B\ x,\ B\ a}(\eta_{B(a)} \circ f))$$

and we propose h as the left inverse. By function extensionality, it's enough to show that for $g : \prod_{a:A} B\ a$,

$$h(\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g) = g$$

By function extensionality we reduce this to

$$h(\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g)\ a = g\ a$$

for $a : A$. First, note that we have

$$h(\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g)\ a \equiv \eta^{-1}_{B\ a}\big(\mathtt{ind}_{\prod_{x:A} B\ x,\ B\ a}(\eta_{B(a)} \circ \mathtt{ev}_a)\ (\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g)\big)$$

By the definition of modalities we have

$$\mathtt{ind}_{\prod_{x:A} B\ x,\ B\ a}(\eta_{B(a)} \circ \mathtt{ev}_a)\ (\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g) = \eta_{B(a)}(\mathtt{ev}_a\ g)$$

so by action on paths we get

$$\eta^{-1}_{B\,a}\big(\mathtt{ind}_{\prod_{x:A} B\ x,\ B\ a}(\eta_{B(a)} \circ \mathtt{ev}_a)\ (\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g)\big) = \eta^{-1}_{B\,a}\big(\eta_{B(a)}(\mathtt{ev}_a\ g)\big)$$

By the definition of a quasi-inverse $\eta^{-1}_{B\,a}(\eta_{B(a)}(\mathtt{ev}_a\ g)) = \mathtt{ev}_a\ g$ which is then equal to $g\ a$. We concatenate the paths needed to reach the desired equality

$$h(\eta^{\bigcirc}_{\prod_{a:A} B(a)}\ g)\ a = g\ a$$

□

Definition 2.0.9. *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 $a : A$.

Nullification at a family of types is an example of a modality, as laid out in [3], where it is presented as a higher inductive type, complete with constructors and eliminators. To avoid listing all these trappings, we will instead look at the important properties $\mathcal{L}_B : \mathcal{U} \to \mathcal{U}$ should have to be considered a nullification operator.

It must have all the data required for it to be a modality.
For all $X : \mathcal{U}$, $\mathcal{L}_B(X)$ must be B-**null**.

Chapter 3

Computability

In this chapter we will give an overview of basic definitions and results of Recursion theory and formalize them in Type theory. The reader is expected to be somewhat familiar with Turing machines, as we will not be delving into their technical details.

Definition 3.0.1. *A **partial function** is a function with a subset of the naturals as its domain.*
Formally

$$f : \Big(\sum_{n:\mathbb{N}} P\ n\Big) \to \mathbb{N}$$

where $P : \mathbb{N} \to \mathbf{Prop}$ *is a family of propositions over* $\mathbb{N}$*, which works as the characteristic function of the domain of the partial function* f*.*

A standard result in Recursion theory is that there exists a Turing machine that enumerates all Turing Machines. We assume a fixed enumeration $T_1, T_2, \ldots$ that we refer to going forward.

Definition 3.0.2. *Church's thesis* (CT) *is an axiom to be assumed, which states that every function* $\mathbb{N} \to \mathbb{N}$ *is computable. Formally in Type theory*

$$\prod_{f:\mathbb{N}\to\mathbb{N}} \Big\| \sum_{e:\mathbb{N}} \prod_{x:\mathbb{N}} \sum_{z:\mathbb{N}} T(e,x,z) \times U(z) = f(x) \Big\|$$

where T *is Kleene's predicate,* e *identifies a Turing machine,* x *is some input to* f *and its corresponding Turing machine* T_e*, finally* z *is the computation history that* T_e *goes through when given* x *as the input, with* $U(z)$ *being the output at the end of the computation.*

In other words, Church's thesis assures us that for every $f : \mathbb{N} \to \mathbb{N}$ there exists some computable function that agrees with it on every input.

Chapter 4

LLPO

In the following definition s is, in places, taken to be an implicit argument. In the sections to follow, terms and types that use it implicitly, are supplied with explicit arguments when the need to be clear arises.

Definition 4.0.1 (LLPO)**.** *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*

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

we can then have by the LLPO a witness for $\mathtt{p_{odd}} \vee \mathtt{p_{even}}$*, where* $\mathtt{p_{odd}}$ *(with* s *as an implicit argument) is the statement that for all odd positions* n*,* $s(n) = 0$*, formally* $\mathtt{p_{odd}} \equiv \prod_{n:\mathbb{N}} (\mathtt{odd}(n) = 1) \to s(n) = 0$*, and* $\mathtt{odd} : \mathbb{N} \to \mathbf{2}$ *with* $\mathtt{odd}\ n = 1$ *iff* n *is odd. Similarly for* $\mathtt{p_{even}}$*.*

We now provide an alternative equivalent formulation of LLPO.

Definition 4.0.2 (LLPO')**.** *LLPO' states that for any* $s : \mathbf{2}^{\mathbb{N}}$ *we have that*

$$\Big(\prod_{n:\mathbb{N}} \big(s\ n = 1 \times \prod_{m:\mathbb{N}} m < n \to s\ m = 0 \big) \to \mathtt{odd}\ n \Big)$$

$$\vee$$

$$\Big(\prod_{n:\mathbb{N}} \big(s\ n = 1 \times \prod_{m:\mathbb{N}} m < n \to s\ m = 0 \big) \to \mathtt{even}\ n \Big)$$

In other words, LLPO' says that for any binary sequence either the first non-zero position is odd, if it exists, or it's even, if it exists.

Lemma 4.0.3. *LLPO is equivalent to LLPO'.*

Proof.

(LLPO $\to$ LLPO') Let $s : \mathbf{2}^{\mathbb{N}}$. Define $\zeta_i : \{1, \dots, i\} \to \mathbf{2}$ by primitive recursion: we define $\zeta_1(1) :\equiv s(1)$. For every $i : \mathbb{N}$, $i > 1$, define ζ_i as such: for $n : \{1, \dots, i\}$ we can decide whether $(n = i) \ + \ (n < i)$. We take cases on this. If $n < i$ then let $\zeta_i(n) :\equiv \zeta_{i-1}(n)$. Otherwise, if $n = i$ then first decide whether $\prod_{m:\{1,\dots,i-1\}} \zeta_{i-1}(m) = 0$ holds or not. If it holds then we let $\zeta_i(i) :\equiv s(i)$. Otherwise we set $\zeta_i(i) :\equiv 0$.

We have effectively defined $\zeta : \prod_{i:\mathbb{N}} \mathbf{2}^{\{1,\dots,i\}} : i \mapsto \zeta_i$. We define $s' : \mathbf{2}^{\mathbb{N}} : n \mapsto \zeta(n)(n)$. One can easily verify that $\texttt{atMost1one}\ s'$. By LLPO we have $\mathtt{p_{odd}}(s') \vee \mathtt{p_{even}}(s')$. Our goal is a mere proposition so we ignore the truncation of the disjunction and take cases on the coproduct. Wlog we only deal with the case $\mathtt{p_{odd}}(s')$. By `inr` our goal is reduced to proving

$$\prod_{n:\mathbb{N}} \big(s\ n = 1 \times \prod_{m:\mathbb{N}} m < n \to s\ m = 0\big) \to \texttt{even}\ n$$

Consider $n : \mathbb{N}$ such that $s\ n = 1$ and for any $m < n$, $s\ m = 0$. We need to show that n is even. We can decide whether $\texttt{even}(n) \ + \ \texttt{odd}(n)$. In the left case we are done. In the right case we have n to be odd. Since for all $m < n$, $s(m) = 0$ and $s(n) = 1$ we have that $\zeta_n(n) = 1$ and by extension $s'(n) = 1$. This contradicts the fact that n is odd and $\mathtt{p_{odd}}(s')$.

(LLPO $\leftarrow$ LLPO') Let $s : \mathbf{2}^{\mathbb{N}}$ such that $\texttt{atMost1one}\ s$. We apply LLPO' to s to get a witness of the consequent disjunction. We drop the truncation and wlog pick the left constituent of the coproduct. So we have

$$p : \prod_{n:\mathbb{N}} \big(s\ n = 1 \times \prod_{m:\mathbb{N}} m < n \to s\ m = 0\big) \to \texttt{odd}\ n$$

We choose to show $\mathtt{p_{even}}\ s$. So, for $n : \mathbb{N}$, $\texttt{even}(n)$, we need to show $s\ n = 0$. We can decide whether $(s\ n = 0) \ + \ (s\ n = 1)$. The first case is trivial. Suppose $s\ n = 1$. We want to reach falsum so that we can resolve the case using Ex Falso. We reduce this to showing n to be odd which comes in contradiction with it being even. $\texttt{odd}(n)$ would follow from p if we only had $\prod_{m:\mathbb{N}} m < n \to s\ m = 0$. So, consider $m : \mathbb{N},\ m < n$. We can decide whether

$(s\ m = 0) + (s\ m = 1)$. In the left case we are done and in the right one we have by $\texttt{atMost1one}(s)$ that $m = n$, but by $m < n$ we can deduce $m \neq n$, contradiction. □

LLPO can be viewed as a weaker form of the Law of Excluded Middle.
Lemma 4.0.4. *The Law of Excluded Middle implies the Lesser Limited Principle of Omniscience.*

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

- $\mathtt{p_{odd}} \Rightarrow$ LLPO, trivially.
- $\neg \mathtt{p_{odd}}$, alongside LEM, implies that there exists odd $n_e : \mathbb{N}$ such that $s(n_e) = 1$. We can now prove $\prod_{n:\mathbb{N}} \mathtt{even}(n) \to s(n) = 0$. Let $n : \mathbb{N}$ such that $\mathtt{even}(n)$ is true. 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 $\texttt{atMost1one}$ we have $n = n_e \Rightarrow n$ even and odd which is a contradiction $c : \bot$. Ex Falso, $\mathtt{efq} : \bot \to s(n) = 0$. Then $\mathtt{efq}(c) : s(n) = 0$.

□

Lemma 4.0.5. *If we replace the consequent of LLPO with its double negation, let's call it LLPO$^{\neg\neg}$, then we can prove it in Type Theory.*

Proof. From LEM $\Rightarrow$ LLPO we have LEM $\Rightarrow$LLPO$^{\neg\neg}$ by effectively the same proof. Since we invoke LEM only twice, we can propose the double negation of these two instances of LEM where the need arises, show that they are provable in our context and then drop the double negation since it's a modality and the goal is of the same modality. We effectively use the same method as when we drop truncations around hypotheses when proving mere propositions, only this time we are working with the double negation modality.

The first instance we come across is

$$\mathtt{p}_{\mathtt{odd}} \vee \neg \mathtt{p}_{\mathtt{odd}}$$

We want to prove $\neg\neg(\mathtt{p}_{\mathtt{odd}} \vee \neg \mathtt{p}_{\mathtt{odd}})$. Assume $q : (\mathtt{p}_{\mathtt{odd}} \vee \neg \mathtt{p}_{\mathtt{odd}}) \to \bot$. We compose q with $|\cdot|$ to get

$$q' : (\mathtt{p}_{\mathtt{odd}} + \neg \mathtt{p}_{\mathtt{odd}}) \to \bot$$

We then have

$$q' \circ \mathtt{inl} : \neg \mathtt{p}_{\mathtt{odd}}$$

and

$$q' \circ \mathtt{inr} : \neg\neg \mathtt{p}_{\mathtt{odd}}$$

which lead us to falsum.

The second instance is

$$\left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\| \vee \neg \left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\|$$

Assume the negation of the above

$$q : \neg \left(\left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\| \vee \neg \left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\| \right)$$

towards contradiction. We compose with $|\cdot|$ to rid ourselves of the truncation

$$q' : \neg \left(\left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\| + \neg \left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\| \right)$$

We then have

$$q' \circ \mathtt{inr} : \neg\neg \left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\|$$

and

$$q' \circ \mathtt{inl} : \neg \left\| \sum_{n_e : \mathtt{Odd}} s(n_e) = 1 \right\|$$

Contradiction. □

Definition 4.0.6. *The Axiom of Countable Choice (ACC) is adapted from its set-theoretic counterpart and states that if for every $n : \mathbb{N}$ there exists (merely) some $b : \|B\ n\|$, where $B : \mathbb{N} \to \mathcal{U}$ is a family of h-sets over $\mathbb{N}$, then there merely exists some $f : \prod_{n:\mathbb{N}} B\ n$.*
More concisely

$$\Big(\prod_{n:\mathbb{N}} \|B\ n\|\Big) \to \Big\|\prod_{n:\mathbb{N}} B\ n\Big\|$$

Lemma 4.0.7. *Under Church's thesis, the following type is inhabited*

$$\sum_{F:\mathbb{N}\to\mathbb{N}\to\mathbf{2}} \Big(\big(\prod_{m:\mathbb{N}} \mathtt{atMost1one}\ F(m)\big) \times \big(\prod_{f:\mathbb{N}\to\mathbf{2}} \big\| \sum_{m,k:\mathbb{N}} F(m, 2k + f\ m) = 1\big\|\big)\Big)$$

Proof. First we provide a witness

$$G : \sum_{F:\mathbb{N}\to\mathbb{N}\to\mathbf{2}} \prod_{n:\mathbb{N}} \mathtt{atMost1one}\ F(n)$$

For $n, m : \mathbb{N}$ we pick the indexed T_n Turing machine of our enumeration. We can decide whether m is odd or even.

- If it's odd, then there exists actual $k : \mathbb{N}$ such that $m = 2k + 1$. If k is the Gödel number of the computation history that T_n goes through when given n as the input, furthermore, if it halts at the end of this computation with output 1 then set $G.\mathtt{fst}\ n\ m :\equiv 1$. Otherwise set $G.\mathtt{fst}\ n\ m :\equiv 0$.
- If it's even, then there exists actual k such that $m = 2k$. We work as in the odd case, with the only difference being that we set $G.\mathtt{fst}\ n\ m :\equiv 1$ iff the output of the halting computation is 0.

Having defined $G.\mathtt{fst}$ it's easy to see that $G.\mathtt{snd}$ has a straightforward proof, which we ommit.

We now define

$$Q : (\mathbb{N} \to \mathbb{N} \to \mathbf{2}) \to \mathcal{U}$$

by

$$Q\ F :\equiv \prod_{f:\mathbb{N}\to\mathbf{2}} \big\| \sum_{m,k:\mathbb{N}} F(m, 2k + f\ m) = 1\big\|$$

We want to prove $Q\ G.\mathtt{fst}$. Consider arbitrary $f : \mathbb{N} \to \mathbf{2}$. By CT we have that there exists, merely, $e : \mathbb{N}$ such that T_e computes f, furthermore there

exists $z : \mathbb{N}$ which z is the Gödel number of the halting computation that T_e goes through when given e as an input and lastly T_e halts at the end of this computation and outputs $j : \mathbb{N}$ where $j = f\ e$. Since it's decidable whether j is 0 or 1, we can take cases on it. In both cases we have $G.\mathtt{fst}(e, 2z + j) = 1$.

We conclude the proof by providing $G.\mathtt{fst}$ as the first component and the product of $G.\mathtt{snd}$ with $Q\ G.\mathtt{fst}$ as the second. □

Theorem 4.0.8.

$$ACC \times CT \times LLPO \to \bot$$

Proof. Let G be the witness we reached in our proof of 4.0.7. Let $F :\equiv G.\mathtt{fst}$. By LLPO we can procure

$$f : \prod_{n:\mathbb{N}} \| \mathtt{p_{odd}}(F\ n) + \mathtt{p_{even}}(F\ n) \|$$

By the ACC we get

$$f' : \Big\| \prod_{n:\mathbb{N}} \mathtt{p_{odd}}(F\ n) + \mathtt{p_{even}}(F\ n) \Big\|$$

Since our goal is $\bot$, which is an h-prop, we can ignore the truncation and act as if we have access to

$$f'' : \prod_{n:\mathbb{N}} \mathtt{p_{odd}}(F\ n) + \mathtt{p_{even}}(F\ n)$$

Let $e : \big(\mathtt{p_{odd}}(F\ n) + \mathtt{p_{even}}(F\ n)\big) \to \mathbf{2}$ be the function that sends $\mathtt{inl}\ \_ : \mathtt{p_{odd}}(F\ n) + \mathtt{p_{even}}(F\ n)$ to 1 and $\mathtt{inr}\ \_$ to 0. It's easy to see that the following holds

$$q : \prod_{n:\mathbb{N}} \big((e \circ f''\ n = 1) \to \mathtt{p_{odd}}(F\ n)\big) \times \big((e \circ f''\ n = 0) \to \mathtt{p_{even}}(F\ n)\big)$$

By $G.\mathtt{snd}$ we have that there exist $m, k : \mathbb{N}$ such that we have

$$l : F(m, 2k + e \circ f''\ m) = 1$$

This should normally be truncated, but given the context, we are allowed to drop it.

We then take cases on $(e \circ f''\ m = 0) + (e \circ f''\ m = 1)$.

- If $e \circ f'' \; m = 0$ then by $q \; m$ we have $\mathtt{p_{even}}(F \; m)$. This contradicts l.
- Similarly, $e \circ f'' \; m = 1$ is also in contradiction with l.

In either case we reach falsum, concluding the proof. □

For those interested in constructive analysis, here's a consequence of LLPO.

Theorem 4.0.9. *LLPO implies* $\prod_{x:\mathbb{R}} x \leq 0 \vee x \geq 0$.

Proof. Let $x : \mathbb{R}$. We define sequence $s_1 : \mathbf{2}^{\mathbb{N}}$ as such: for $n : \mathbb{N}$, x_n is a rational. So we can decide $(x_n < -\frac{1}{n}) \; + \; (x_n \geq -\frac{1}{n})$. In the left case define $s_1 \; n :\equiv 1$ and in the right case $s_1 \; n :\equiv 0$. Similarly define s_2 so that for $n : \mathbb{N}$ decide $(x_n > \frac{1}{n}) \; + \; (x_n \leq \frac{1}{n})$ and let $s_2 \; n :\equiv 1$ if $x_n > \frac{1}{n}$ and $s_2 \; n :\equiv 1$ if $x_n \leq \frac{1}{n}$. Define $s : \mathbf{2}^{\mathbb{N}}$ by interleaving s_1 and s_2, so that $s(2n+1) :\equiv s_1(n)$ and $s(2n) :\equiv s_2(n)$. We apply LLPO' to s and take cases on the resulting coproduct, after dropping the truncation. Wlog we deal only with

$$p : \prod_{n:\mathbb{N}} \big(s \; n = 1 \times (\prod_{m:\mathbb{N}} m < n \to s \; m = 0)\big) \to \mathtt{odd} \; n$$

We will try to prove $x \leq 0$ which can then be followed by `inl` to conclude the main goal. So we need to show that $\prod_{n:\mathbb{N}} x_n \leq \frac{1}{n}$. Let $l : \mathbb{N}$. x_l is a rational so we have that $(x_l \leq \frac{1}{l}) \; + \; (x_l > \frac{1}{l})$. We take cases on this coproduct. The left case is trivial.

In the right case, we have $s_2 \; l = 1$ which implies $s \; 2l = 1$. We will construct a witness, by induction on $\mathbb{N}$, of

$$\prod_{n:\mathbb{N}} \Bigg(\Big(\sum_{m:\mathbb{N}} \big((m < n) \times (s \; m = 1) \times (\prod_{k:\mathbb{N}} k < m \to s \; k = 0)\big) \Big) + \big(\prod_{m:\mathbb{N}} m < n \to s \; m = 0\big) \Bigg)$$

which we will use to prove our current goal, by arriving at a contradiction. We name our witness to be constructed f. $f(0)$ is trivial to construct by showing that no natural is bellow 0 and using `inr`. We assume that we have some $f(n)$ for $n : \mathbb{N}$ and want to define $f(\texttt{succ}\ n)$. We take cases on $f(n)$. In the left case we have an actual m less than n such that $s(m) = 1$ and s is constantly zero bellow it. By definition $n < \texttt{succ}\ n$ so $m < \texttt{succ}\ n$. This fact along with $s\ m = 1$ and the constantness of s to 0 bellow m can be combined into a triplet to which we apply `inl` to get our definition of $f(\texttt{succ}\ n)$.

In the right case we have $\prod_{m:\mathbb{N}} m < n \to s\ m = 0$. Note that we can decide $(s\ n = 0)\ +\ (s\ n = 1)$. We take cases on this. In the left case we simply apply `inr` to the fact that s is constantly 0 bellow $\texttt{succ}\ n$ and we are done. In the right we can use n as the candidate for the triplet and then apply `inl` to the triplet.

We are done with defining f. We now want to construct a witness $k : \sum_{n:\mathbb{N}}(s\ n = 1) \times \prod_{m:\mathbb{N}} m < n$. We take cases on $f\ 2l$. The left case is trivial. In the right case we have that for every $n : \mathbb{N}$ less than $2l$, $s\ n = 0$ and we already have $s\ 2l = 1$ as a hypothesis, these two facts are enough to conclude the case. By $p(k)$ we have that $k.\texttt{fst}$ is odd. So there exists natural m such that $k.\texttt{fst} = 2m + 1$. We can decide whether $(x_m < -\frac{1}{m})\ +\ (x_m \geq -\frac{1}{m})$ and we do take cases on it.

First we look at the right case where $x_m \geq -\frac{1}{m}$. This implies that $s_1(m) = 0$ and by extension $s(k.\texttt{fst}) = s(2m+1) = 0$. Contradiction, since $s(k.\texttt{fst}) = 1$. In the left case we have $x_m < -\frac{1}{m}$. But then $\texttt{abs}(x_l - x_m) > \frac{1}{l} + \frac{1}{m}$ this is in contradiction with our assumption that x is a real number. $\square$

Chapter 5

UTT + CT + MP + LLPO

In this chapter we present the main result of this text, which is showing that Univalent Type Theory is consistent with Church's thesis, Markov's principle and LLPO. Markov's Principle is the following statement

$$\Big(\prod_{n:\mathbb{N}}(P\ n + \neg P\ n)\Big) \to \Big(\neg\prod_{n:\mathbb{N}}\neg P\ n\Big) \to \Big\|\sum_{n:\mathbb{N}} P\ n\Big\|$$

where $P : \mathbb{N} \to \texttt{Prop}$ is a family of propositions over $\mathbb{N}$. Informally, Markov's Principle states that if we have a collection of decidable propositions and the fact that not all of them are false, then one of them must be true.

To reach the consistency result, we will work with models and results presented in [4]. First, some quick terminology. A cwf (category with families) model of type theory is a categorical construction which 'realizes' all data of formal TT presented in 1.7. A detail of their construction that is of interest to us, is that any type over some context is 'realized' in the cwf model as the set of its terms over the same context.

A quick overview of the overarching proof goes like this:
Start with a model of ETT+CT+MP. Show that this model satisfies a statement that will go by the name of IP'. Construct, based on this model, another model of UTT+CT+MP+IP'. Identify the null - with respect to LLPO - types of this model to retrieve a model of UTT+CT+MP+LLPO.
We proceed with the actual proof.

As the first step towards our goal, we consider the cwf model built on top of category of internal cubical objects in $\mathbf{Asm}(\mathcal{K}_1)$. For the category of assemblies over Kleene's first model, $\mathbf{Asm}(\mathcal{K}_1)$, we direct the reader to Chapter 1 of [2]. We call this model $\mathcal{E}$. By Theorem 3.10 of [4] $\mathcal{E}$ satisfies Assumption 3.1 of the same paper. So it is a model of extensional type theory. Furthermore, Church's Thesis and Markov's Principle hold in it, as shown in the proof of theorem 6.4. of [4]. should i argue more about this?

In addition to the above, $\mathcal{E}$ validates the following 'Independence Principle' i claim, but don't I have to rationalize this?

$$\left(\prod_{s:\mathbf{2}^{\mathbb{N}}} P\ s \to \left(\prod_{n:\mathbb{N}} s\ n = 0\right) \to \left\| \sum_{z:\mathbb{N}} Q\ s\ z \right\|\right)$$

$$\to \left(\prod_{s:\mathbf{2}^{\mathbb{N}}} P\ s \to \left\| \sum_{z:\mathbb{N}} \left(\prod_{n:\mathbb{N}} s\ n = 0\right) \to Q\ s\ z \right\|\right)$$

where $P : \mathbf{2}^{\mathbb{N}} \to \mathtt{Prop}$ and $Q : \mathbf{2}^{\mathbb{N}} \to \mathbb{N} \to \mathtt{Prop}$ are families of propositions. Putting it plainly, if the left hand side of the above is true, then $z : \mathbb{N}$ does not depend on the proof of the constantness of s to 0.

Definition 5.0.1. *Given $f : C \to D$ we say that it's constant if for any $c_1, c_2 : C$, $f(c_1) = f(c_2)$.*
Formally,

$$\mathtt{isConst}\ f :\equiv \prod_{c_1,c_2:C} f(c_1) = f(c_2)$$

Remark 5.0.2. Let $A :\equiv \sum_{a:\mathbf{2}^{\mathbb{N}}} \mathtt{atMost1one}\ a$. When referring to elements of A we implicitly mean the first part of the pair. Let

$$B : A \to \mathcal{U}$$

$$B :\equiv \lambda\ a.\ \mathtt{p_{odd}}\ a + \mathtt{p_{even}}\ a$$

We set as our new subgoal to reach an Orton-Pitts model, as seen in Chapter 3 of [4], which models UTT, CT and MP. We call this model to be constructed $\mathcal{E}'$. Furthermore, we require that $\mathbb{N}$ is $\|B\|$-null in this model. To make sure that this last requirement is met, we would like the following instance of IP to hold in it.

$$\prod_{a:A} \prod_{h:\sum_{k:B\ a\to\mathbb{N}} \mathtt{isConst}(k)} \Bigg($$

$$\Big(\prod_{s:\mathbf{2}^{\mathbb{N}}} \Big(\big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n)\big) \vee \big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n+1)\big)\Big) \to \Big(\prod_{n:\mathbb{N}} s\ n = 0\Big) \to \Big\| \sum_{z:\mathbb{N}} \prod_{p:B\ a} k\ p = z \Big\|\Big)$$

$$\to \Big(\prod_{s:\mathbf{2}^{\mathbb{N}}} \Big(\big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n)\big) \vee \big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n+1)\big)\Big) \to \Big\| \sum_{z:\mathbb{N}} \Big(\prod_{n:\mathbb{N}} s\ n = 0\Big) \to \prod_{p:B\ a} k\ p = z \Big\|\Big)\Big)$$

The fact that $\mathbb{N}$ is $\|B\|$-null, follows from this in Intensional Type Theory. Our plan of action shall be to compromise and find a consequent of it, IP', weaker than IP in ETT but strong enough to imply $\|B\|$-nullness of $\mathbb{N}$ in UTT, that has the right form so that by Theorem 6.1 of [4] we get our $\mathcal{E}'$ that satisfies IP'. Note that we have been and will be working in ETT up until the construction of $\mathcal{E}'$ is finalized. This is because we are proving results pertinent to $\mathcal{E}$.

We drop the truncation around $\|\sum_{z:\mathbb{N}} \prod_{p:B\ a} k\ p = z\|$ and since this is the consequent of the antecedent, the resulting statement is weaker than the original IP. We then uncurry 4 times to reach what we propose as our IP', a function that takes four arguments in the form of a quaternary dependent pair

$$
\begin{aligned}
a :&\quad A\\
h :&\quad \sum_{k:B\ a\to\mathbb{N}} \mathtt{isConst}(k)\\
&\quad \Big(\prod_{\bar{s}:\mathbf{2}^{\mathbb{N}}} \big((\prod_{n:\mathbb{N}} \bar{s}(n) = a(2\cdot n)) + (\prod_{n:\mathbb{N}} \bar{s}(n) = a(2\cdot n+1))\big) \to\\
\_ :&\quad \Big(\prod_{n:\mathbb{N}} \bar{s}\ n = 0\Big) \to \sum_{\bar{z}:\mathbb{N}} \prod_{p:B\ a} k\ p = \bar{z}\Big)\\
r :&\quad \sum_{s:\mathbf{2}^{\mathbb{N}}} \big((\prod_{n:\mathbb{N}} s(n) = a(2\cdot n)) + (\prod_{n:\mathbb{N}} s(n) = a(2\cdot n+1))\big)
\end{aligned}
$$

and has return type

$$\Big\|\sum_{z:\mathbb{N}} \Big(\prod_{n:\mathbb{N}} s\ n = 0\Big) \to \prod_{p:B\ a} k\ p = z\Big\|$$

The observant reader should have noticed that the disjunctions in the third and fourth arguments have been replaced with coproducts. This seemingly makes the unnamed third argument weaker which would have the undesirable effect of potentially strengthening IP' beyond IP. Luckily the consequent of $\_$ is a mere proposition which means that swapping disjunction for coproduct does not actually change the strength of the argument. In the case of r the argument becomes stronger, which is satisfactory in itself.

So IP' holds in $\mathcal{E}$, since IP holds in it and is stronger than IP'. We need the following lemma and its corollary to bundle together CT, MP and IP'.

Lemma 5.0.3. *Consider families of types B_1, B_2 where $B_i : A_i \to \mathcal{U}$ and $A_i : \mathcal{U}$, $i \in \{1, 2\}$.*

Define $B : A_1 + A_2 \to \mathcal{U}$, $B(\mathtt{inl}\ a) :\equiv B_1(a)$ for $a : A_1$ and $B(\mathtt{inr}\ a) :\equiv B_2(a)$ for $a : A_2$. Then the following type

$$\prod_{a:A_1} \|B_1\ a\| \times \prod_{a:A_2} \|B_2\ a\|$$

is equivalent to

$$\prod_{a:A_1+A_2} \|B\ a\|$$

Proof. ($\Rightarrow$) Suppose that we have $p : \prod_{a:A_1} \|B_1\ a\| \times \prod_{a:A_2} \|B_2\ a\|$. We want to construct a function $\prod_{a:A_1+A_2} \|B\ a\|$. We use induction on the coproduct and wlog work out only the case $\prod_{a:A_1} \|B(\mathtt{inl}\ a)\|$. For $a : A_1$ we have that $p.\mathtt{fst}(a)$ is a witness of $\|B(\mathtt{inl}\ a)\|$.

($\Leftarrow$) Suppose $q : \prod_{a:A_1+A_2} \|B\ a\|$. We want to construct a witness for $\prod_{a:A_1} \|B_1\ a\| \times \prod_{a:A_2} \|B_2\ a\|$. We do so by constructing witnesses for both of the constituents of the product. Wlog we do that only for $\prod_{a:A_1} \|B_1\ a\|$. Let $a : A_1$. Then $q(\mathtt{inl}\ a) : \|B_1\ a\|$, by the definition of B. □

Corollary 5.0.4. *The above lemma generalizes from the case of two families of types B_1, B_2 to a finite collection of families $B_1, \ldots, B_n$.*

This fact enables us to reformulate a finite collection of statements, each one in the correct form, to a single statement in that same form which is equivalent to their conjunction. So, given that CT, MP and IP' are in the right form required by Theorem 6.1 of [4], by the corollary above and said theorem, we have that there exists Orton-Pitts model $\mathcal{E}'$ of UTT, CT, MP and IP'.

Proposition 5.0.5. *$\mathcal{E}'$ is an Orton-Pitts model of UTT in which CT, MP and IP' hold.*

5.1 Null types in $\mathcal{E}'$

The proofs in this section are all about $\|B\|$-nullness i.e. equivalence between function types (see Definition 2.0.9). We work in the context of Intensional Type Theory. Our approach in showing that some $F : C \to D$ is a map with contractible fibers, is to provide for any $d : D$ (merely) some $c : C$ such that $f(c) = d$ and then show that such a c is unique (its type is a proposition), where C, D are said function types.

Lemma 5.1.1. *$\mathbb{N}$ is $\|B\|$-null in $\mathcal{E}$.*

Proof. Given $a : A$ and $f : \|B\ a\| \to \mathbb{N}$ we need to prove that there exists unique $f' : \mathbf{1} \to \mathbb{N}$ through which f factors, in the sense that $f = f' \circ g$, where $g : \|B\ a\| \to \mathbf{1}$ is the sole inhabitant of its function space. Functions with $\mathbf{1}$ as their domain are constant. We therefore need to find some $z : \mathbb{N}$ so that $\lambda\ \_.\ z$ is f'. To that end, we will use IP' which we've proven true in

$\mathcal{E}$. We invoke it twice. In both times, the first three arguments shall be the same. The choice for the first argument, a, is evident.

We provide the composition $f \circ |\cdot| : B\ a \to \mathbb{N}$ as k of second argument. We need to provide a witness for the second part of h, $\texttt{isConst}(\texttt{f} \circ |\cdot|)$. Let $q_1, q_2 : B\ a$. We need to show that $f\ |q_1| = f\ |q_2|$. Since $\|B\ a\|$ is a proposition, we have $|q_1| = |q_2|$ and then by action on paths we get the desired equality.

We need to construct a witness for the third argument. Let $\bar{s} : \mathbf{2}^{\mathbb{N}}$ such that $\bar{s}$ is constantly equal to 0 and at the same time it is *actually* equal to the odd subsequence of a or to the even one i.e. $\big(\prod_{n:\mathbb{N}} \bar{s}(n) = a(2 \cdot n)\big) + \big(\prod_{n:\mathbb{N}} \bar{s}(n) = a(2 \cdot n + 1)\big)$. We take cases on this coproduct. Wlog suppose that $\prod_{n:\mathbb{N}} \bar{s}(n) = a(2 \cdot n)$. We can use this to construct a witness $\texttt{evenSubseqIsZero} : B\ a$. We then put forward $f\ |\texttt{evenSubseqIsZero}|$ as $\bar{z}$. Recall that we've set k to be $f \circ |\cdot|$. We need to show that for arbitrary $p : B\ a$, we have $f\ |p| = f\ |\texttt{evenSubseqIsZero}|$. This follows from $\|B\ a\|$ being a proposition and action on paths.

Finally, for argument r we provide the odd s_1 and even s_2 subsequences of a, along with proofs that they are indeed subsequences of a and we get hold of $z_1 : \mathbb{N}$ and $z_2 : \mathbb{N}$ respectively, along with

$$\zeta_i : \Big(\prod_{n:\mathbb{N}} s_i\ n = 0\Big) \to \prod_{p:B\ a} \big(f(|p|) = z_i\big)$$

for $i \in \{1, 2\}$. Recall that the current goal is a mere proposition, namely that merely exists some f', that is why we can act as if we have actual z_1 and z_2. Equality on $\mathbb{N}$ is decidable, therefore $(z_1 = z_2) + (z_1 \neq z_2)$ is provable.

We will first prove that the existence of a candidate for f' follows from both constituents of the coproduct.

- First the case where $z_1 = z_2$. We arbitrarily pick z_1 and propose $f' :\equiv \lambda\ \_.\ z_1 : \mathbf{1} \to \mathbb{N}$. By function extensionality we reduce proving $f = f' \circ g$ to proving $f\ c = f'(g\ c)$ for arbitrary $c : \|B\ a\|$.
  Since our goal is a mere proposition ($\mathbb{N}$ is a set, so equality on it is a proposition), we can act as if we have access to an actual $b : B\ a$. We take cases on b.
  - In the first case we have $\mathtt{p}_{\mathtt{odd}}$ which trivially leads us to $b_1 : \prod_{n:\mathbb{N}} s_1 = 0$. We then have $\zeta_1(b_1)(b) : f(|b|) = z_1$. Since $\|B\ a\|$ is

a mere proposition, we have $|b| = c$. By action on paths on this and f we get $f\ c = z_1$. By definition, $f'(g(c)) = z_1$, which we concatenate with $f(c) = z_1$ to get $f\ c = f'(g\ c)$ and conclude this case.

  - In the case where $\mathtt{p_{even}}$, we similarly construct $b_2 : \prod_{n:\mathbb{N}} s_2 = 0$. We then have $\zeta_2(b_2)(b) : f(|b|) = z_2$. The proof follows closely the previous case, only this time around we have to include $z_1 = z_2$ in the concatenation of paths.

- Now consider the case $z_1 \neq z_2$. For any $n : \mathbb{N}$, $a\ n = 1$ is decidable. Suppose that $\prod_{n:\mathbb{N}} a\ n \neq 1$. Then a and by extension s_1 and s_2 are constantly 0. We use these facts to construct $b : B\ a$ and $b_i : \prod_{n:\mathbb{N}} s_i = 0$ and then concatenate $\zeta_i(b)(b_i)$ like before to get $z_1 = f\ b = z_2$. This contradicts $z_1 \neq z_2$. We have proven $\neg \prod_{n:\mathbb{N}} a\ n \neq 1$. By Markov's Principle there exists $n_1 : \mathbb{N}$ such that $a\ n_1 = 1$. We pick the subsequence s_0 with parity opposite to n_1 and let f' be always equal to the corresponding z_0. We prove $\prod_{n:\mathbb{N}} s_0\ n = 0$ by induction on $\mathbb{N}$ and cases on $(s_0\ n = 0) + (s_0\ n = 1)$. In the case where $s_0\ n = 1$ we can reach falsum because n will have to be both even and odd (in a) by `atMost1one` a. Ex falso trivializes this case. Now that we have established this too, we can construct $b : B\ a$ and have enough arguments for IP' to output the desired equality $f\ b = z_0$ which proves homotopy between f and $(\lambda\ \_.\ z_0) \circ g$.

We still need to prove the uniqueness of f'. Since $\mathbf{1}$ is finite and equality on $\mathbb{N}$ is decidable, we can decide equality on the function space

$$p : \prod_{f_1, f_2 : \mathbf{1} \to \mathbb{N}} f_1 = f_2 + f_1 \neq f_2$$

We will use this to prove that $\sum_{h:\mathbf{1}\to\mathbb{N}} f = h \circ g$ is a mere proposition, which entails the uniqueness of f'. Consider $h_1, h_2 : \mathbf{1} \to \mathbb{N}$ such that $f = h_i \circ g,\ i \in \{1, 2\}$. By p we have $(h_1 = h_2) + (h_1 \neq h_2)$. We use induction on the coproduct. The left case is trivial. For the right case we have $q : h_1 \neq h_2$ and we will first try to prove $B\ a \to \bot$ as a stepping stone. Consider $b : B\ a$. Since $h_1 \circ g = f = h_2 \circ g$, by homotopy $h_1(g\ b) = h_2(g\ b)$. We can then prove $\prod_{*:\mathbf{1}} h_1(*) = h_2(*)$. By function extensionality $h_1 = h_2$. But this contradicts q, so we reach $\bot$. We've just proven $\tau : B\ a \to \bot$. We

still have to prove $h_1 = h_2$, the main goal. By 4.0.5 we have $(B\ a \to \bot) \to \bot$ and by τ we get $\bot$. We conclude the proof by Ex Falso. □

Corollary 5.1.2. *Similarly,* $\mathbf{2}$ *is* $\|B\|$*-null in* $\mathcal{E}$*.*

Lemma 5.1.3. *In* $\mathcal{E}$ *if* C, D *are* $\|B\|$*-null then so is* $C + D$*.*

Proof. We need to show that for $f : \|B\ a\| \to C + D$ there exists unique $f' : \mathbf{1} \to C+D$ such that $f = f' \circ e$, where $e : \|B\ a\| \to \mathbf{1}$ is the sole inhabitant of its function space. We will first show that a candidate f' merely exists and follow that up with a proof of uniqueness.

We define

$$g : C + D \to \mathbf{2}$$

using components

$$g_C :\equiv \lambda\ \_.\ 0 : C \to \mathbf{2}$$

$$g_D :\equiv \lambda\ \_.\ 1 : D \to \mathbf{2}$$

Recall that in $\mathcal{E}$ we have access to a witness of IP' which takes the following arguments in the form of a 4-product

$$\begin{aligned}
a :&\quad A\\
h :&\quad \sum_{\bar{f}:B\ a\to C+D} \mathtt{isConst}(\bar{f})\\
_ :&\quad \Big(\prod_{\bar{s}:\mathbf{2}^{\mathbb{N}}} \Big(\big(\prod_{n:\mathbb{N}} \bar{s}(n) = a(2\cdot n)\big) + \big(\prod_{n:\mathbb{N}} \bar{s}(n) = a(2\cdot n+1)\big)\Big) \to\\
&\qquad \Big(\prod_{n:\mathbb{N}} \bar{s}\ n = 0\Big) \to \sum_{z:\mathbf{2}} \prod_{b:B\ a} g(\bar{f}\ b) = z\Big)\\
r :&\quad \sum_{s:\mathbf{2}^{\mathbb{N}}} \Big(\big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n)\big) + \big(\prod_{n:\mathbb{N}} s(n) = a(2\cdot n+1)\big)
\end{aligned}$$

and returns

$$\Big\| \sum_{z:\mathbf{2}} \Big(\prod_{n:\mathbb{N}} s\ n = 0\Big) \to \prod_{b:B\ a} g(\bar{f}\ b) = z \Big\|$$

We provide this with the arguments required, two times, where $\bar{f}$ is f composed with the truncation map $|\cdot| : B\ a \to \|B\ a\|$ and $\mathtt{isConst}(\bar{f})$ follows from composition with $|\cdot|$.

For the unnamed third argument we work as in the proof of 5.1.1. Suppose that we have $\bar{s} : \mathbf{2}^{\mathbb{N}}$ equal to the odd or even subsequence of a and always equal to 0. We take cases on whether it's equal to the odd or even subsequence. Wlog suppose it's equal to the odd one. We use `inl` on this fact to get $b_0 : B\ a$. We then propose $g(\bar{f}\ b_0)$ as z. We still need to show that for any $b : B\ a$ we have $g(\bar{f}\ b) = g(\bar{f}\ b_0)$. Since $\bar{f} \equiv f \circ |\cdot|$ and $|b| = |b_0|$, by action on paths we have $\bar{f}\ b = \bar{f}\ b_0$. We use action on paths again to reach the desired equality $g(\bar{f}\ b) = g(\bar{f}\ b_0)$.

For the fourth argument we use s_1 the odd subsequence of a the first time and s_2 the even one the second time.

In return, we get a witnesses

$$z_i : \mathbf{2}$$

and

$$\zeta_i : \Big(\prod_{n:\mathbb{N}} s_i\ n = 0\Big) \to \prod_{b:B\ a} g(f\ |b|) = z_i$$

for $i \in \{1, 2\}$. Note that we dropped the truncation since our goal is a mere proposition, therefore we can act as if we have an actual z_1, z_2 and ζ_1, ζ_2.

We shift our attention to proving a new subgoal in the form of

$$\Big\| \sum_{z:\mathbf{2}} \prod_{b:\|B\ a\|} g(f\ b) = z \Big\|$$

By decidability of equality of naturals we have that $z_1 = z_2 + z_1 \neq z_2$. We take cases on this.

- First the case where $z_1 = z_2$. Wlog propose z_1 as z. We have to show that $\prod_{b:\|B\ a\|} g(f\ b) = z_1$. Let $b : \|B\ a\|$. Equality on $\mathbf{2}$ is a proposition, so the current goal $g(f\ b) = z_1$ is a proposition. We can therefore act as if we have some $b' : B\ a$. We take cases on this. Wlog we only deal with the case where the odd subsequence s_1 is constant to 0. We cast ζ_1 upon this fact to conjure a proof of $\prod_{b:B\ a} g(f\ |b|) = z_1$. So $g(f\ |b'|) = z_1$. Since $|b'| = b$, by action on paths we have $g(f\ b) = g(f\ |b'|)$. We concatenate the two paths to reach $g(f\ b) = z_1$ and conclude the case.
- Now the case where $z_1 \neq z_2$. We copy the following passage almost verbatim from earlier in the chapter.

For any $n : \mathbb{N}$, $a\ n = 1$ is decidable. Suppose that $\prod_{n:\mathbb{N}} a\ n \neq 1$. Then a and by extension s_1 and s_2 are constantly 0. We use these facts to construct $b : B\ a$ and $b_i : \prod_{n:\mathbb{N}} s_i = 0$ and then concatenate $\zeta_i(b)(b_i)$ like before to get $z_1 = f\ b = z_2$. This contradicts $z_1 \neq z_2$. We have proven $\neg \prod_{n:\mathbb{N}} a\ n \neq 1$. By Markov's Principle there exists $n_1 : \mathbb{N}$ such that $a\ n_1 = 1$.

Now, let s_0 be the subsequence opposite of the parity of n_1. We want to show that $\prod_{n:\mathbb{N}} s_0\ n = 0$. Let $n : \mathbb{N}$. It's decidable whether $s_0\ n = 0$ or $s_0\ n = 1$. The first case is a direct proof while the second leads to a contradiction with $\texttt{atMost1one}(a)$ and 1 appearing in both subsequences, which we can resolve by Ex Falso. Now that we have s_0 to be constant to 0, we propose the corresponding z_0 as z and apply the corresponding ζ_0 to get a proof of $\prod_{b:B\ a} g(f\ |b|) = z_0$. From $\prod_{n:\mathbb{N}} s_0\ n = 0$ we can immidiately get $b_0 : B\ a$. So we have $g(f\ |b_0|) = z_0$. For any $b : \|B\ a\|$ we have $|b_0| = b$ and by action on paths we get $g(f\ b) = g(f\ |b_0|)$. We concatenate with the above to get $g(f\ b) = z_0$ and conclude the case.

We've proven

$$\bar{y} : \left\| \sum_{z:\mathbf{2}} \prod_{b:\|B\ a\|} g(f\ b) = z \right\|$$

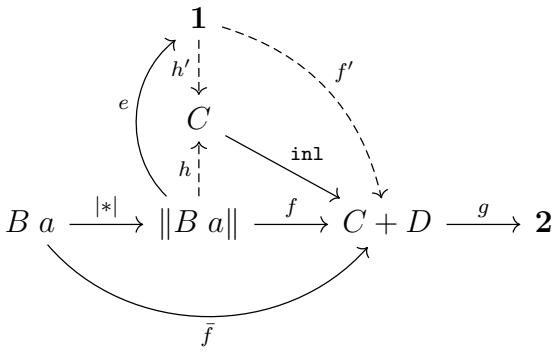
Given that the current goal, which is proving the mere existence of candidate f', is a proposition, we can ignore the truncation and work with

$$y : \sum_{z:\mathbf{2}} \prod_{b:\|B\ a\|} g(f\ b) = z$$

We can prove that

$$y = 0 + y = 1$$

We use induction on this coproduct to prove our goal. Without loss of generality, we argue only for the case of $y = 0$. We would like to have an $h : \|B\ a\| \to C$ such that $\texttt{inl} \circ h = f$.



To that end we will first try to construct

$$\bar{h} : \prod_{b:\|B\ a\|} \sum_{c:C} \mathtt{inl}(c) = f(b)$$

Let $b : \|B\ a\|$. First, note that we can prove

$$\xi : \prod_{a:C+D} \Big(\big(\sum_{c:C} \mathtt{inl}\ c = a \big) + \big(\sum_{d:D} \mathtt{inr}\ d = a \big) \Big)$$

by induction on $C + D$. We then take cases on

$$\xi(f\ b) : \big(\sum_{c:C} \mathtt{inl}\ c = f\ b \big) + \big(\sum_{d:D} \mathtt{inr}\ d = f\ b \big)$$

- If $\bar{c} : \sum_{c:C} \mathtt{inl}\ c = f\ b$, we define $\bar{h}\ b :\equiv \bar{c}$.
- If $\sum_{d:D} \mathtt{inl}\ d = f\ b$, then $g(f\ b) = 1$ but since $y = 0$ we also have $g(f\ b) = 0$. Contradiction. This case is resolved by Ex Falso.

Now that we have $\bar{h}$ in our hands, we will use it to construct h which should satisfy $\mathtt{inl} \circ h = f$ as stated earlier. Let $b : \|B\ a\|$. Define $h\ b :\equiv \mathtt{fst}(\bar{h}\ b)$. Now, in order to show that $\mathtt{inl} \circ h = f$, we just show, by function extensionality, that $\mathtt{inl}(h\ b) = f\ b$. This follows directly from the definitions of h and $\bar{h}$.

Since C is $\|B\|$-null, there exists $h' : \mathbf{1} \to C$ such that $h = h' \circ e$. Observe that $\mathtt{inl} \circ h' : \mathbf{1} \to C + D$, furthermore $(\mathtt{inl} \circ h') \circ e = f$. We have found a valid candidate for f', what is left is to show it's unique.

Consider any f' candidate. We can construct a $f'_C : \mathbf{1} \to C$ so that f' factors through it, $f' = \mathtt{inl} \circ f'_C$, with our construction being similar to that of h

earlier. We want to show that $f' = \mathtt{inl} \circ h'$ or $\mathtt{inl} \circ f'_C = \mathtt{inl} \circ h'$. We will show that $h' = f'_C$. Since C is $\|B\|$-null, $\sum_{k:\mathbf{1}\to C} k \circ e = h$ is a mere proposition. Therefore showing $h' = f'_C$ is reduced to showing $f'_C \circ e = h$. We use function extensionality. We know that for arbitrary $b : \|B\ a\|$, $\mathtt{inl}(f'_C \circ e\ b) = \mathtt{inl}(h\ b)$. By 2.12.1 of [5] we have $(\mathtt{inl}\ a_1 = \mathtt{inl}\ a_2) \simeq (a_1 = a_2)$. We can then deduce that $f'_C \circ e\ b = h\ b$. This concludes the proof of uniqueness of h'.

□

Lemma 5.1.4. *In $\mathcal{E}$ if C is $\|B\|$-null then for any $c_1, c_2 : C$, the identity type $c_1 =_C c_2 : \mathcal{U}$ is $\|B\|$-null.*

Proof. Follows immediately from the fact that $\mathcal{L}_{\|B\|}$ is a modality and the fourth datum of Definition 2.0.1. □

5.2 CT + LLPO + UTT

We form a null types model $\mathcal{E}'_{\|B\|}$ as defined in Section 5 Definition 5.1 of [4] based on $\mathcal{E}'$ from before. The types of this model are the $\|B\|$-null types of $\mathcal{E}$. The witnesses of these types are also inherited from $\mathcal{E}'$. Same for contexts. We claim that this model of UTT satisfies CT, MP and LLPO. To prove this claim we first need to procure some tools.

Remark 5.2.1. When we say that the types and witnesses carry over, we mean that if $\Gamma \vdash_{\mathcal{E}'} A : \mathcal{U}$ and A is $\|B\|$-null in $\mathcal{E}'$ then $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} [\![A]\!]^{\mathcal{E}'} : \mathcal{U}$ where $[\![A]\!]^{\mathcal{E}'}$ is the construct in $\mathcal{E}'$ that *realizes* $\Gamma \vdash_{\mathcal{E}'} A : \mathcal{U}$.

Lemma 5.2.2. *The types $\mathbb{N}$, $\mathbf{2}$, $\mathbf{1}$ of $\mathcal{E}'$ carry over to $\mathcal{E}'_{\|B\|}$ and are each equal to their corresponding $\mathbb{N}$, $\mathbf{2}$, $\mathbf{1}$ of $\mathcal{E}'_{\|B\|}$ in $\mathcal{E}'_{\|B\|}$. Similarly, if C is a $\|B\|$-null type in $\mathcal{E}'$ then identity types of witnesses of it, carry over to $\mathcal{E}'_{\|B\|}$ and are equal to their correspondent there. Same for dependent products and sums, if in $\mathcal{E}'$ $C : D \to \mathcal{U}$ such that $C(d)$ is $\|B\|$-null for all $d : D$ then $\prod_{d:D} C(d)$ carries over to $\mathcal{E}'_{\|B\|}$ and equals its correspondent and if in addition C is $\|B\|$-null too in $\mathcal{E}'$, then $\sum_{d:D} C(d)$ carries over and equals its correspondent too.*

Proof. The types $\mathbb{N}$ and $\mathbf{2}$ are $\|B\|$-null in $\mathcal{E}'$ as shown in 5.1.1 and its corollary. It's trivial to show that the unit type $\mathbf{1}$ is $\|B\|$-null too. $\|B\|$-nullness of dependent products and sums under the assumptions above, follow from

2.0.8 and the fact that nullification is a modality and therefore forms a $\sum$-closed reflective subuniverse as seen in 2.0.4. $\|B\|$-nullness of identity types follows once again from the fact that nullification is a modality and the fourth datum of 2.0.1.

We've proven that these types carry over to $\mathcal{E}'_{\|B\|}$. What is left, is to show that they are equal to their corresponding construction in $\mathcal{E}'_{\|B\|}$, i.e. in the case of the dependent product we would have to show that for $C : D \to \mathcal{U}$, where C and D are types in $\mathcal{E}'_{\|B\|}$, the type $[\![\prod_{d:D} C(d)]\!]^{\mathcal{E}'}$ in $\mathcal{E}'_{\|B\|}$ is equal to the local $\prod_{d:D} C(d)$ in $\mathcal{E}'_{\|B\|}$.

Since we are working under univalence, equality follows from equivalence. We can prove this using the data of these inductive types, constructors like `succ`, 0 and the induction principles, which all carry over to $\mathcal{E}'_{\|B\|}$ since their types are $\|B\|$-null in $\mathcal{E}'$. We omit the details on how done and direct those looking for more exposition on how this should be carried out to Section 5.4 of [5] which talks about universal properties of inductive types. □

It should be mentioned that the above applies to coproducts too, whose case we work out in detail, to elucidate the more general case above.

Example 5.2.3. Let $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} C : \mathcal{U}$, $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} D : \mathcal{U}$ and $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} C + D : \mathcal{U}$. We have that

$$\Gamma \vdash_{\mathcal{E}'_{\|B\|}} [\![C + D]\!]^{\mathcal{E}'} : \mathcal{U}$$

and

$$\Gamma \vdash_{\mathcal{E}'_{\|B\|}} p : \big((C + D) = [\![C + D]\!]^{\mathcal{E}'}\big)$$

Proof. Since C, D are types in $\mathcal{E}'_{\|B\|}$, they must be $\|B\|$-null types in $\mathcal{E}'$. By 5.1.3 we have that $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} [\![C + D]\!]^{\mathcal{E}'} : \mathcal{U}$. It only remains to show that $[\![C + D]\!]^{\mathcal{E}'}$ is equal to $C + D$ in $\mathcal{E}'_{\|B\|}$. Since this is a UTT model, by univalence it's enough to show that the types are equivalent. The coproduct data of $[\![C + D]\!]^{\mathcal{E}'}$ in $\mathcal{E}'$ are available in $\mathcal{E}'_{\|B\|}$, since their types must be $\|B\|$-null by 2.0.8. We name them $\mathtt{ind}_1$, $\mathtt{inl}_1$ and $\mathtt{inr}_1$ to distinguish them from those of $\Gamma \vdash_{\mathcal{E}'_{\|B\|}} C + D : \mathcal{U}$ whose we denote with $\mathtt{ind}_2$, $\mathtt{inl}_2$ and $\mathtt{inr}_2$. It's easy to verify that

$$(\mathtt{ind}_2\ \mathtt{inl}_1\ \mathtt{inr}_1) : (C + D) \to [\![C + D]\!]^{\mathcal{E}'}$$

and

$$(\mathtt{ind}_1\ \mathtt{inl}_2\ \mathtt{inr}_2) : [\![C + D]\!]^{\mathcal{E}'} \to (C + D)$$

form a pair of quasi-inverses and conclude the proof of equivalence. □

Theorem 5.2.4. *CT, MP and LLPO hold in the model $\mathcal{E}'_{\|B\|}$ of UTT.*

Proof. We first work out the proof for CT. CT is the following type

$$\prod_{f:\mathbb{N}\to\mathbb{N}} \left\| \sum_{e:\mathbb{N}} \prod_{x:\mathbb{N}} \sum_{z:\mathbb{N}} T(e,x,z) \times U(z) = f(x) \right\|$$

which, after adopting the alias CT for the above type, would be the construct $\vdash_{\mathcal{E}'_{\|B\|}} \mathrm{CT} : \mathcal{U}$ in the model $\mathcal{E}'_{\|B\|}$. Note that T and U are both functions from $\mathbb{N}$ to $\mathbb{N}$, so the results in 5.2.2 extend to them. With this in mind, along with the general result of 5.2.2 and Corollary 5.6 of [4], we can deduce, by induction on type formation, that there is a path p in $\mathcal{E}'_{\|B\|}$ between

$$\vdash_{\mathcal{E}'_{\|B\|}} \mathrm{CT} : \mathcal{U}$$

and the type

$$\vdash_{\mathcal{E}'_{\|B\|}} \left[\!\!\left[\prod_{f:\mathbb{N}\to\mathbb{N}} \mathcal{L}_{\|B\|} \left\| \sum_{e:\mathbb{N}} \prod_{x:\mathbb{N}} \sum_{z:\mathbb{N}} T(e,x,z) \times U(z) = f(x) \right\| \right]\!\!\right]^{\mathcal{E}'} : \mathcal{U}$$

which is the carried over $\|B\|$-null type in $\mathcal{E}'$

$$\vdash_{\mathcal{E}'} \prod_{f:\mathbb{N}\to\mathbb{N}} \mathcal{L}_{\|B\|} \left\| \sum_{e:\mathbb{N}} \prod_{x:\mathbb{N}} \sum_{z:\mathbb{N}} T(e,x,z) \times U(z) = f(x) \right\| : \mathcal{U}$$

So we need only find a witness of this last type in $\mathcal{E}'$ which would carry over to the same type in $\mathcal{E}'_{\|B\|}$ and could be transported along p to conclude the case of CT. This type can be proved to follow from the interpretation of CT in $\mathcal{E}'$

$$\vdash_{\mathcal{E}'} \mathrm{CT} : \mathcal{U}$$

which we know to be inhabited by 5.0.5.

For MP we follow exactly the same steps as for CT.

For LLPO we first reduce proving

$$\vdash_{\mathcal{E}'_{\|B\|}} \mathrm{LLPO} : \mathcal{U}$$

to providing a witness for

$$a : A \vdash_{\mathcal{E}'_{\|B\|}} \|B\ a\| : \mathcal{U}$$

where LLPO $\equiv \prod_{a:A} \|B\ a\|$ for A and B presented in 5.0.2. Then work as in the case for CT and prove a path p between

$$a : A \vdash_{\mathcal{E}'_{\|B\|}} \|B\ a\| : \mathcal{U}$$

and

$$a : A \vdash_{\mathcal{E}'_{\|B\|}} [\![\mathcal{L}_{\|B\|}\ \|B\ a\|]\!]^{\mathcal{E}'} : \mathcal{U}$$

By Corollary 5.5 of [4] we have a witness of the last type. We transport it over p to conclude the proof. □

Appendix

Following [1] we develop a constructive theory of rational and real numbers.

5.3 Rational numbers

We define the rationals as $\mathbb{Q} :\equiv \sum_{n,m:\mathbb{I}} m \neq 0$ where m is the denominator and n the numerator. We now need to codify the usual notion of equality of rationals into type theory. We define it as the infix predicate

$$\doteq :\equiv \prod_{q_1,q_2:\mathbb{Q}} (q_1.\mathtt{fst} \cdot q_2.\mathtt{snd}) = (q_2.\mathtt{fst} \cdot q_1.\mathtt{snd})$$

which can be proved to be an equivalence relation. This sort of equality between rationals is the only one that interests us and at no point will we be refering to the type theoretic primitive equality between $\mathbb{Q}$ witnesses, without explicit mention that we are doing so.

Similarly for order we use

$$q_1 < q_2 :\equiv (q_1.\mathtt{fst} \cdot q_2.\mathtt{snd}) < (q_2.\mathtt{fst} \cdot q_1.\mathtt{snd})$$

where the $<$ in the right hand side of the definition is order defined on the naturals. As expected, we define $\leq$ as the coproduct of $<$ and $\doteq$.

Defining the usual operations on rationals and showing that the usual relations hold is straightforward.

5.4 Real numbers

We define the reals as Cauchy sequences of rationals

$$\mathbb{R} :\equiv \sum_{s:\mathbb{Q}^{\mathbb{N}}} \prod_{n,m:\mathbb{N}} \Big(\mathtt{abs}\big(s(n) - s(m)\big) \leq n^{-1} + m^{-1}\Big)$$

where `abs` is the absolute value on rationals. We also define `abs` on a real number x to be the maximum out of x and $-x$, $\mathtt{max}(x, -x)$, where `max` is defined component-wise, $\mathtt{max}(x, y)_n :\equiv \mathtt{max}(x_n, y_n)$.

Addition and subtraction on reals are also defined component-wise, $(x \pm y)_n :\equiv x_{2n} + y_{2n}$. We won't be needing multiplication so we skip it entirely. Rationals are embedded in the reals as constant sequences of themselves, since if $q : \mathbb{Q}$ then $r_q :\equiv \lambda \_.\ q$ is clearly a valid first component of a real number.

We codify equality on reals as such

$$\doteq :\equiv \prod_{x,y:\mathbb{R}} \prod_{n:\mathbb{N}} \mathtt{abs}\big(x.\mathtt{fst}(n) - y.\mathtt{fst}(n)\big) \leq \frac{2}{n}$$

This, like the rational dot equality, is what we mean when talk about equality between reals, not the type theoretic primitive.

We define what it means for a real number to be **positive**

$$\mathtt{positive} :\equiv \prod_{x:\mathbb{R}} \sum_{n:\mathbb{N}} x.\mathtt{fst}(n) > \frac{1}{n}$$

We define the order predicate '$<$' on reals with infix notation

$$x < y \ :\equiv \mathtt{positive}\big(\mathtt{abs}(y - x)\big)$$

We define $\leq$ as follows

$$x \leq y :\equiv \prod_{n:\mathbb{N}} \big(y.\mathtt{fst}(n) - x.\mathtt{fst}(n)\big) \geq -\frac{1}{n}$$

Bibliography

[1] Douglas Bridges and Fred Richman. "THE FOUNDATIONS OF CONSTRUCTIVE MATHEMATICS". In: *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1987. DOI: 10.1017/CBO9780511565663.002.

[2] Jaap van Oosten. *Realizability, Volume 152: An Introduction to Its Categorical Side*. San Diego, USA: Elsevier Science, 2008. ISBN: 0444515844, 9780444515841.

[3] Egbert Rijke, Michael Shulman, and Bas Spitters. *Modalities in homotopy type theory*. 2017. eprint: arXiv:1706.07526.

[4] Andrew Swan and Taichi Uemura. *On Church's Thesis in Cubical Assemblies*. 2019. eprint: arXiv:1905.03014.

[5] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.