Modeling set theory in homotopy type theory

Jérémy Ledent
Supervised by Bas Spitters & Freek Wiedijk
Radboud University of Nijmegen, Netherlands

May 19 to August 08, 2014

Abstract

Homotopy type theory is a new foundation of mathematics under current development. To compare it with the existing set theoretic foundation, we formalize the cumulative hierarchy of sets in the COQ system, closely following and clarifying the informal treatment in the homotopy type theory book [17].

Introduction

The univalent foundations project [17] is developing new foundations of mathematics motivated by a recently discovered connection between homotopy theory and type theory. Through this connection, they offer new insight into both those subjects, as well as into higher category theory. These foundations are based on homotopy type theory (HoTT), a variant of Martin-Löf type theory augmented by the so-called axiom of Univalence and the possibility to define higher inductive types.

In order to have faith in these new foundations, we want to compare them to the old set-theoretic foundations. Such comparisons between type theories and constructive set theories have already been studied by P. Aczel [1], A. Miquel [10], B. Barras [3] and many others. These investigations allow us to measure the proof-theoretic strength of the different theories, which is very useful since it allows us to have good intuitions on how much can be proved in them. For example, Zermelo set theory (Z) is stronger than Peano Arithmetic (PA) as we can build a model of PA in Z. Hence, Z proves the consistency of PA. Similarly, we can build a model of Z in ZF, or show that ZF and ZFC are equiconsistent. The proof theoretic strength of Martin-Löf type theory has been studied by A. Setzer [13]. Knowing the strength of HoTT is still an open problem.

On the one hand, HoTT can be modelled within ZFC + a hierarchy of Grothendieck universes, using the so-called Kan simplicial sets [8]. Conversely, if we add the axiom of choice and classical logic to HoTT, we can model ZFC by the cumulative hierarchy V, defined as a higher inductive type (see Theorem 2.7). However, by adding choice and excluded middle to HoTT, we lose one of its main features: constructiveness. In this report, we try to understand what can be done without these principles. Thus, we are interested in relating homotopy type theory and constructive versions of set theory such as CZF or its impredicative counterpart, IZF [2].

We have formalized our results using the Coq proof assistant [9]. As noted in the HoTT book, the cumulative hierarchy V is a non-standard higher inductive type, so it is a good test case for formalization. Hence, this is a good experiment to better understand higher inductive types. The Coq development is available at `https://github.com/jledent/vset`. We have submitted a pull request to add it to the Coq HoTT library.

Our main contribution is the formalization of the results of §10.5 of the HoTT book [17]. This work allowed us to find a mistake in the induction principle of V given in the book: a solution was found in collaboration with Peter Lumsdaine and Mike Shulman, and we prove that it is correct (see section 3.2). We also did some theoretical investigations, which led to the definition in V of the set of hereditarily finite sets, $H(\omega)$ (see section 4.2).

In this report, we assume the reader is familiar with the basics of dependent type theory, and knows a little bit about Coq. The first section introduces some basic notions of homotopy type theory. The second section defines the cumulative hierarchy V, a model of set theory in HoTT, following §10.5 of the HoTT book [17]. Section 3 deals with my formalization of these results in Coq, and the last section discusses some investigations on the links with constructive set theories.

1 Homotopy type theory

In this section, I will present a short introduction to homotopy type theory. It is obtained by adding the univalence axiom (section 1.4) and higher inductive types (section 1.3) to Martin-Löf type theory:

$$\text{HoTT} = \text{MLTT} + \text{Univalence} + \text{Higher Inductive Types}$$

First, I will recall some facts about Martin-Löf type theory, assuming the reader has some basic knowledge on dependent types. Then, I will explain the notions of homotopy type theory that are relevant to my development. More details on the subject can be found in the HoTT book [17]. As in the HoTT book, I will adopt a rather informal style.

In particular I will avoid giving the full syntax of terms and types or the associated inference rules (they can be found in the appendix of the HoTT book).

1.1 Martin-Löf intensional type theory

Martin-Löf Type Theory (MLTT) is a constructive foundation of mathematics which is based on the Curry-Howard correspondence. There are two primitive judgements in MLTT:

- Typing derivations $t : A$ indicate that the term t has type A. We can interpret types as propositions, and terms of a certain type as proofs of the associated proposition.
- Judgemental equality $x \equiv y$ that can be understood as the fact that that the terms x and y are equal by definition. This judgement also includes computation rules such as β-reduction.

This version of MLTT is called *intensional* because of the distinction between judgemental equality and propositional equality (see below). An *extensional* type theory, where both notions of equality coincide, would be incompatible with the axiom of univalence (see 1.4).

Type Universes. In MLTT, types are terms, and hence they themselves must have a type. However, having a "type of all types" would make our system inconsistent, similarly to how a "set of all sets" causes Russell's paradox in set theory [7].

To avoid this issue, we assume given an infinite hierarchy of *universes*: $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$ We assume that for every i, $\mathcal{U}_i : \mathcal{U}_{i+1}$, and moreover if $A : \mathcal{U}_i$, then $A : \mathcal{U}_{i+1}$.

Most of the time, the level i of the universes will be left implicit, but in order to check that a proof is valid, we must verify that there is a consistent way of assigning universe levels. In Coq, the type universes are called Type and the indexes are assigned automatically.

Curry-Howard correspondence for MLTT. We briefly recall the type formers of MLTT, and their intuitive meaning with respect to the Curry-Howard correspondence. In this paragraph, A and B are types, and $P : A \to \mathcal{U}$ is a type family (that is, for every $x : A$, $P(x)$ is a type).

- *Product type* $A \times B$: this corresponds to the logical "and". A proof of $A \times B$ consists of a pair (a, b) where a is a proof of A, and b is a proof of B.

- *Sum type* $A+B$: this is the logical "or". It can be constructed in two ways, either from a proof of A or from a proof of B. It differs from the usual "or" in that it contains information on which of the two propositions A and B is true.
- *Dependent function type* $\prod_{(x:A)} P(x)$: this corresponds to the "for all" quantifier. It is a function that maps every $x : A$ to a proof of $P(x)$. When the family P is constant (i.e., $P(x) \equiv B$ for all x), it is written $A \to B$.
- *Dependent pair type* $\sum_{(x:A)} P(x)$: this is the "there exists" quantifier. A proof of $\sum_{(x:A)} P(x)$ consists of a pair (x, t) where $x : A$ is a witness and t is a proof of $P(x)$. Hence, unlike for the usual existential quantifier, it contains a particular witness x which satisfies P.

In section 1.2, definition 1.6, we will define the notion of *mere propositions*, which are types that behave in the usual mathematical way. We can then equip them with the traditional logical connectives $\exists$ and $\vee$.

Propositional equality. For any type A and $x, y : A$, we define the *identity type* $x =_A y$ as an inductive type with one constructor:

$$\mathsf{refl} : \prod_{x:A} (x =_A x)$$

When there is no ambiguity, we will omit the type and simply write $x = y$.

Its associated induction principle is called *path induction* in view of the homotopy interpretation of section 1.2. It says that given a type family $C : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$, in order to construct a function $f : \prod_{(x,y:A)} \prod_{(p:x=y)} C(x, y, p)$, we only need to give its value when y is x and p is refl_x. That is, we must provide a term $c : \prod_{(x:A)} C(x, x, \mathsf{refl}_x)$. The computation rule says that the function f thus defined verifies $f(x, x, \mathsf{refl}_x) \equiv c(x)$.

Intuitively, path induction says that in order to prove a statement which is quantified over every x, y, and every proof of equality $p : x = y$, we can assume that p is reflexivity.

1.2 The homotopical interpretation of type theory

In type theory, one often wants the so-called *axiom K* or, equivalently, the *Uniqueness of Identity Proofs (UIP)*, which says that $\prod_{(x,y:A)} \prod_{(p,q:x=y)}(p = q)$, i.e., two proofs of $x = y$ are always equal. These two axioms have the consequence that the structure of identity types becomes trivial: the only possible proof of equality is reflexivity.

M. Hofmann and T. Streicher showed that K is not provable in intensional type theory by exhibiting a model in which it is false [6]. This model, which is based on groupoids (i.e., categories in which all arrows are invertible), can be seen as a first step towards homotopy type theory. Indeed, Hofmann and Streicher observed that in this model isomorphic types can be identified, and they call it “universe extensionality”. This principle can be seen as a weak form of univalence.

Recently, V. Voevodsky proposed a new model of type theory using simplicial sets [8, 16], in which axiom K is also false. The structure of identity types that arises from this model is the starting point of the homotopical interpretation of type theory.

In homotopy type theory, types are viewed as homotopy types, i.e., topological spaces up to homotopy. An element of a type is interpreted as a point in that space, a proof of equality $p : x = y$ is a path from x to y, $h : p =_{(x=y)} q$ is a homotopy (or 2-path) from p to q, and so on. According to Grothendieck's homotopy hypothesis, we can equivalently see types as ∞-groupoids where the objects are the terms of that type, and the n-morphisms are the n-paths.

The ∞-groupoid structure of types. Propositional equality can be proven to be symmetric and transitive (and reflexive by definition). Formally, we have the following lemmas, which are easily proven by path induction:

Lemma 1.1 (Symmetry). *For every $A : \mathcal{U}$ and $x, y : A$, there is a function which to $p : x = y$ associates its inverse $p^{-1} : y = x$, such that $\mathsf{refl}_x^{-1} \equiv \mathsf{refl}_x$*

Lemma 1.2 (Transitivity). *For every $A : \mathcal{U}$ and $x, y, z : A$, there is a function which to $p : x = y$ and $q : y = z$ associates their concatenation $p \cdot q : x = z$, such that $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$.*

These two functions can be shown to verify the usual laws that we would expect: $p \cdot p^{-1} = \mathsf{refl}_x$, the inverse is involutive, concatenation is associative, etc. This structure also extends to higher paths, which is why types can indeed be interpreted as ∞-groupoids.

Type families. According to the Curry-Howard correspondence, a type family $P : A \to \mathcal{U}$ corresponds to a property of the elements of A. Using path induction, we can prove the *indiscernibility of identicals*, which is the fact that if x and y are equal, then they must satisfy the same properties. In the homotopical interpretation, type families are fibrations, and from this perspective the indiscernibility of identicals will be called *transport*.

Lemma 1.3 (Transport). *For every type family $P : A \to \mathcal{U}$ and path $p : x = y$, there is a function $\mathsf{transport}^P(p, -) : P(x) \to P(y)$.*

We can now define the notion of dependent paths:

Definition 1.4. Let $P : A \to \mathcal{U}$, $x, y : A$, $p : x = y$, $u : P(x)$ and $v : P(y)$. The type of *dependent paths* from u to v lying over p is $\mathsf{transport}^P(p, u) = v$, and is noted $u =_p^P v$ (see fig. 1).

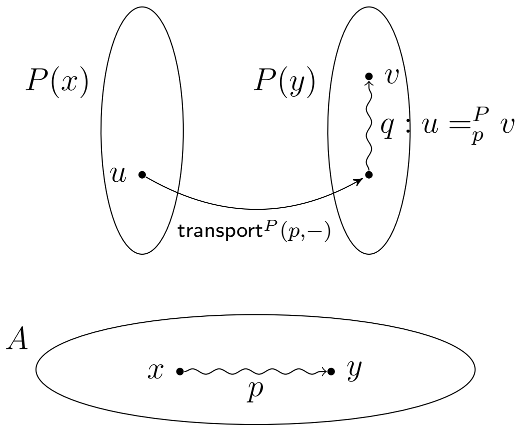

Figure 1: q is a dependent path lying over p

Lemma 1.5. *We can apply functions to paths:*

- *Given* $f : A \to B$, *there is a function* $\mathsf{ap}_f : \prod_{(p:x=y)} \big(f(x) = f(y)\big)$
- *Given* $f : \prod_{(x:A)} P(x)$, *there is a function* $\mathsf{apd}_f : \prod_{(p:x=y)} \big(f(x) =_p^P f(y)\big)$

Proof. By path induction. □

The hierarchy of h-levels. We can establish a hierarchy of types based on how complex their groupoid structure is.

Definition 1.6.
- A *mere proposition* is a type A which is proof-irrelevant, that is, such that the type $\mathsf{isProp}(A) :\equiv \prod_{(x,y:A)}(x = y)$ is inhabited.
- An *h-set* is a type A all of whose identity types are mere propositions. Equivalently, this means that the type $\mathsf{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)}(p = q)$ is inhabited.
- More generally, a type is n-truncated if its identity type is $(n-1)$-truncated. Mere propositions correspond to (-1)-truncated types, and h-sets to 0-truncated ones.

For a fixed $\mathcal{U}$, we can also define the sub-universes of n-truncated types in $\mathcal{U}$. Here, we will be mostly interested in $\mathsf{hProp}_{\mathcal{U}} :\equiv \sum_{(A:\mathcal{U})} \mathsf{isProp}(A)$, sometimes noted hProp, and $\mathsf{hSet}_{\mathcal{U}} :\equiv \sum_{(A:\mathcal{U})} \mathsf{isSet}(A)$, sometimes noted hSet. Thus, an inhabitant of hProp

(respectively, hSet) consists of a type A along with a proof that it is a mere proposition (respectively, an h-set). hProp should not be confused with Coq's "Prop" universe, which is impredicative and therefore not used for HoTT.

Intuitively, mere propositions are types which contain no more information than whether or not they are inhabited. These types are interesting because they allow us to model the usual mathematical notion of propositions. Indeed, unlike their set-theoretical counterparts "or" and "exists", the "sum" and "sigma" types contain additional information. For example, a proof of $\sum_{(x:A)} P(x)$ contains a particular witness x which satisfies P. This is not the case in usual mathematics. Thus, we add to our type theory a new type constructor which allows us to truncate a type by removing all information except its inhabitedness.

Definition 1.7. For any type A, the *propositional truncation* of A, denoted $\|A\|$, is a type such that:

- For all $a : A$, there is $|a| : \|A\|$
- $\|A\|$ is a mere proposition

This type can be constructed using higher inductive types (see 1.3 and §6.9 of [17]). Thanks to it, we can now define the *traditional logical notations*:

$$\begin{aligned} P \wedge Q &:\equiv P \times Q \\ P \vee Q &:\equiv \|P + Q\| \\ \forall (x : A).\, P(x) &:\equiv \prod_{x:A} P(x) \\ \exists (x : A).\, P(x) &:\equiv \left\| \sum_{x:A} P(x) \right\| \end{aligned}$$

The $\wedge$ and $\forall$ connectives are the same as their type-theoretic counterparts, but the $\vee$ and $\exists$ are truncated in order to "forget" the additional information.

1.3 Higher inductive types

Higher inductive types are a generalization of the inductive types which are already present in Martin-Löf type theory. They allow us to define new types which are freely generated by some constructors. The difference with regular inductive types is that those constructors are allowed to generate not only points in the type being defined, but also paths and higher paths.

When we define a type by higher induction, we also get an associated induction principle, which is analogous to the induction principle of a regular inductive type, with an additional condition to prove for each path constructor.

Giving a general definition of higher inductive paths is still a subject of current research, see for example [14]. Here, we will only explain what such a type looks like with a simple example, the type of the circle. Other examples include the truncation type (discussed previously in definition 1.7), and set-quotients (see [17], §6.10).

Example 1.8 (The circle $\mathbb{S}_1$)**.** Let us consider the higher inductive type $\mathbb{S}_1$, which is generated by the following two constructors:

- $\mathsf{base} : \mathbb{S}_1$
- $\mathsf{loop} : \mathsf{base} = \mathsf{base}$

The first constructor base is a *point constructor*, and the second one, loop, is called a *path constructor*. This means that $\mathbb{S}_1$ consists of a point base and a path loop from base to itself. The fact that a higher inductive type is *freely* generated by its constructors is reflected in the fact that loop is a new inhabitant of the identity type, a priori not equal to $\mathsf{refl}_{\mathsf{base}}$.

Obtaining the non-dependent induction principle of a higher inductive type is quite straightforward. Given a type B with the same structure as $\mathbb{S}_1$, there is a map from $\mathbb{S}_1$ to B which maps the constructors to that structure. More formally, given a type B, a point $b : B$ and a path $\ell : b = b$, there is a $f : \mathbb{S}_1 \to B$ such that $f(\mathsf{base}) = b$ and $\mathsf{ap}_f(\ell) = \mathsf{loop}$.

In the dependent case, the idea is similar, but we have to be careful with the paths constructors. Given a type family $P : \mathbb{S}_1 \to \mathcal{U}$, we would like to construct a dependent function $f : \prod_{(x:\mathbb{S}_1)} P(x)$. The dependent induction principle of $\mathbb{S}_1$ says that, to construct f, we only need to fulfil the following two conditions:

- give an element $b : P(\mathsf{base})$.
- give a dependent path $\ell : b =^P_{\mathsf{loop}} b$.

Similarly to how b must be a point in the fiber over base, the dependent path ℓ must be lying over the corresponding constructor loop. The induction principle of $\mathbb{S}_1$ also tells us that the dependent function f thus constructed is such that $f(\mathsf{base}) = b$ and $\mathsf{apd}_f(\mathsf{loop}) = \ell$.

1.4 Univalence

Voevodsky's *Univalence axiom* is one of the two big novelties of homotopy type theory (along with higher inductive types, see section 1.3). One of its consequences is that isomorphic structures can be identified, which is common practice in mathematics but usually labelled as an "abuse of notation".

Equivalences. In homotopy type theory, equivalences play the role of isomorphisms between types. If there exists an equivalence $f : A \to B$, we say that A and B are are *equivalent* and write $A \simeq B$.

Definition 1.9. A map $f : A \to B$ is an *equivalence* if there is $g : B \to A$ such that

$$\big((f \circ g \sim \mathsf{Id}_B) \times (g \circ f \sim \mathsf{Id}_A)\big)$$

where $f \sim f'$ denote $\prod_{(x:A)}(f(x) = f'(x))$.

Remark: This definition of equivalence is actually incompatible with the axiom of univalence (introduced below), because it is not a mere proposition. To fix this, it is for example possible to add an additional "half-adjoint" condition. The different possible definitions of equivalence are discussed in §4.1 to §4.5 of the HoTT book [17]. For the purpose of this report, this definition will be sufficient.

Lemma 1.10. *For all $A, B : \mathcal{U}$, there is a function* $\mathsf{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B)$

Proof. Given $p : A = B$, we can build a function $f : A \to B$ by transporting p along the type family $X \mapsto X$. To prove that f is an equivalence we proceed by path induction: if p is refl_A, then f is the identity function id_A, which is an equivalence. □

Axiom 1.11 (Univalence)**.** The map $\mathsf{idtoeqv}$ is an equivalence. Therefore:

$$(A = B) \simeq (A \simeq B)$$

In particular, this gives us a new way of constructing paths, since we have a function $(A \simeq B) \to (A = B)$.

Example 1.12. We can give an equivalent definition of the circle (see example 1.8): Let $\mathbb{S}'_1$ be the higher inductive type generated by the following four constructors:

- $\mathsf{N}, \mathsf{S} : \mathbb{S}'_1$
- $\mathsf{east}, \mathsf{west} : \mathsf{N} = \mathsf{S}$

Univalence allows us to show that $\mathbb{S}_1 = \mathbb{S}'_1$, by showing that they are isomorphic.

Proof. Using the induction principle of $\mathbb{S}_1$, we can define $f : \mathbb{S}_1 \to \mathbb{S}'_1$ such as $f(\mathsf{base}) = \mathsf{S}$ and $\mathsf{ap}_f(\mathsf{loop}) = \mathsf{east}^{-1} \cdot \mathsf{west}$. Conversely, we define $g : \mathbb{S}'_1 \to \mathbb{S}_1$ by $g(\mathsf{N}) = \mathsf{base}$, $g(\mathsf{S}) = \mathsf{base}$, $\mathsf{ap}_g(\mathsf{east}) = \mathsf{loop}$ and $\mathsf{ap}_g(\mathsf{west}) = \mathsf{refl}_{\mathsf{base}}$. Remains to verify that f is indeed an equivalence with inverse g (see definition 1.9), which is easily checked. □

Example 1.13. We can use univalence to construct non-trivial paths: consider the function $f : \mathsf{bool} \to \mathsf{bool}$ defined by $f(\mathsf{True}) :\equiv \mathsf{False}$ and $f(\mathsf{False}) :\equiv \mathsf{True}$. We can easily verify that f is an equivalence, hence $\mathsf{bool} \simeq \mathsf{bool}$, and by univalence, we get a path $p : \mathsf{bool} = \mathsf{bool}$. Then assuming that $p = \mathsf{refl}_{\mathsf{bool}}$, we can show that f is the identity function and hence $\mathsf{True} = \mathsf{False}$, which leads to a contradiction.

Thus, the axiom of univalence is incompatible with axiom K.

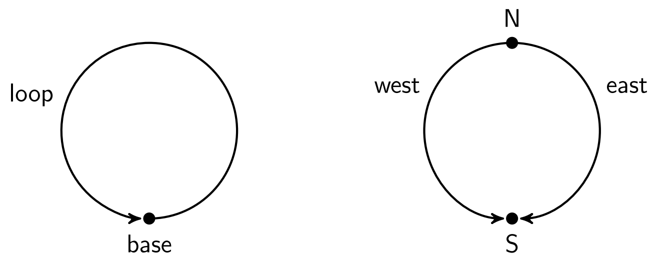

Figure 2: $\mathbb{S}_1$ (left) and $\mathbb{S}_1'$ (right)

2 The cumulative hierarchy V

2.1 Definition and induction principle

Let $\mathcal{U}$ be a fixed universe. We want to define the *cumulative hierarchy* V of all sets in $\mathcal{U}$, such that V itself is again a set, living in a higher universe $\mathcal{U}'$. We are going to define it as a higher inductive type.

The basic idea is the following:

- Given a type $A : \mathcal{U}$ and a function $f : A \to V$, we can construct $\mathsf{set}(A, f) : V$ which denotes the image of A under f (in set-theoretic language, that would be $\{ f(a) \mid a \in A \}$).
- To ensure that this holds, we must make $\mathsf{set}(A, f)$ and $\mathsf{set}(B, g)$ equal whenever A and B have the same image under f and g, i.e. when:

  $$\mathsf{Eq\_img}(f, g) :\equiv \big(\forall (a : A).\, \exists (b : B).\, f(a) = g(b)\big) \wedge \big(\forall (b : B).\, \exists (a : A).\, f(a) = g(b)\big)$$

  This is done using a path constructor.
- Finally, since we want V itself to be an h-set, we add the 0-truncation constructor.

Examples The hierarchy V is bootstrapped from the empty map $\mathsf{rec}_{\mathbf{0}}(V) : \mathbf{0} \to V$, which gives the empty set $\varnothing = \mathsf{set}(\mathbf{0}, \mathsf{rec}_{\mathbf{0}}(V))$. Then we can build the singleton $\{\varnothing\} : V$ using the function $\star \mapsto \varnothing$ from the unit type to V, and so on.

Thus, what we would like to define is a type having the following structure:

Definition 2.1. A type V is a *cumulative hierarchy* if there exists three terms set, setext and $\mathsf{0\text{-}trunc}$ of the following types:

(i) $\mathsf{set} : \prod_{(A:\mathcal{U})} (A \to V) \to V$

(ii) $\mathsf{setext} : \prod_{(A,B:\mathcal{U})} \prod_{(f:A\to V)} \prod_{(g:B\to V)} \mathsf{Eq\_img}(f, g) \to \mathsf{set}(A, f) = \mathsf{set}(B, g)$

(iii) $\mathsf{0\text{-}trunc} : \prod_{(x,y:V)} \prod_{(p,q:x=y)} (p = q)$

We would also like V to be equipped with an appropriate induction principle. The obvious idea would be to define V as a higher inductive type, with three constructors set, setext and 0-trunc. However, higher inductive types are not fully understood yet, and there is no general description of how they behave. As discussed in the HoTT book [17], this definition of V wouldn't fit in the framework of the currently well-understood higher inductive types.

The reason why this definition would be non-standard is that two of the constructors, namely setext and 0-trunc, take as input not only points but also *paths* in V. In the case of setext, the hypothesis $\mathsf{Eq\_img}(f, g)$ even contains truncations of sums of paths in V. Constructors that take as input points in the type being defined are not problematic: when we construct a function by induction on that type, we recursively assume that the function has already been defined at those points. However, when the constructors take as input recursive occurrences of paths, it is not entirely clear what the associated hypothesis of the induction principle should be.

Fortunately in this case, there is a workaround which allows us to avoid dealing with this issue. We are going to use a different encoding of V in order to have a more standard definition whose induction principle is straightforward. The workaround for the 0-trunc constructor is treated in details in the HoTT book [17] (§ 6.9), so we will only discuss setext here.

Definition 2.2. A *bitotal* relation $R : A \to B \to \mathsf{hProp}$ is such that:

$$\mathsf{Bitot}(R) :\equiv \big(\forall (a : A).\, \exists (b : B).\, R(a, b)\big) \wedge \big(\forall (b : B).\, \exists (a : A).\, R(a, b)\big)$$

Definition 2.3. Let $R : A \to B \to \mathsf{hProp}$ be a relation. The *R-pushout* $A \sqcup^R B$ is the higher inductive type generated by:

(i) $\mathsf{i} : A \to A \sqcup^R B$

(ii) $\mathsf{j} : B \to A \sqcup^R B$

(iii) $\mathsf{glue} : \prod_{(a:A)} \prod_{(b:B)} R(a, b) \to \mathsf{i}(a) = \mathsf{j}(b)$

We can now give a proper definition of V:

Definition 2.4. The type of *V-sets* V is the higher inductive type generated by:

(i) $\mathsf{set} : \displaystyle\prod_{A:\mathcal{U}} (A \to V) \to V$

(ii) $\mathsf{setext}' : \displaystyle\prod_{(A,B:\mathcal{U})} \prod_{(R:A\to B\to \mathsf{hProp})} \mathsf{Bitot}(R) \to \prod_{h:A\sqcup^R B\to V} \big(\mathsf{set}(A, h\circ \mathsf{i}) = \mathsf{set}(B, h \circ \mathsf{j})\big)$

(iii) $\mathsf{0\text{-}trunc}' : \prod_{f:\mathbb{S}'_1 \to V} (\mathsf{ap}_f(\mathsf{west}) = \mathsf{ap}_f(\mathsf{east}))$ where $\mathbb{S}'_1$ is defined in example 1.12

We can check that with this definition, V is indeed a cumulative hierarchy in the sense of definition 2.1.

Theorem 2.5. *The type V of definition 2.4 is a cumulative hierarchy.*

Proof. We must provide the three terms set, setext and $\mathsf{0\text{-}trunc}$. set is exactly the point constructor of V, and the case of $\mathsf{0\text{-}trunc}$ is treated in [17]. Let us see how to construct setext.

Given $A, B : \mathcal{U}$, $f : A \to V$ and $g : B \to V$, along with a proof $r : \mathsf{Eq\_img}(f, g)$, we want a path $p : \mathsf{set}(A, f) = \mathsf{set}(B, g)$. We can define $R(a, b) :\equiv \big(f(a) = g(b)\big)$, so that $\mathsf{Bitot}(R)$ is exactly $\mathsf{Eq\_img}(f, g)$. Then, the induction principle of $A \sqcup^R B$ gives us a function $h : A \sqcup^R B \to V$ such that $\begin{cases} h \circ \mathsf{i} \equiv f \\ h \circ \mathsf{j} \equiv g \end{cases}$ (i.e., the diagram on fig. 2.1 commutes). Thus, $\mathsf{setext}'(A, B, R, r, h)$ is the path we are looking for.

$$\begin{tikzcd}
& A \arrow[d, "\mathsf{i}"] \arrow[rdd, bend left, "f"] & \\
B \arrow[r, "\mathsf{j}"] \arrow[rrd, bend right, "g"'] & A \sqcup^R B \arrow[rd, "h"] & \\
& & V
\end{tikzcd}$$

Figure 3: Factoring f and g through $A \sqcup^R B$

□

Induction principle of V Since we have defined V as a higher inductive type, we also get an associated induction principle, which is the following. Given $P : V \to \mathcal{U}$, along with:

(i) $H_{\mathsf{set}} : \prod_{(A:\mathcal{U})} \prod_{(f:A\to V)} \big(\prod_{a:A} P(f(a))\big) \to P(\mathsf{set}(A, f))$

(ii) $H_{\mathsf{setext}'} : \prod_{(A,B:\mathcal{U})} \prod_{(R:A\to B\to\mathcal{U})} \prod_{(r:\mathsf{Bitot}(R))} \prod_{(h:A\sqcup^R B\to V)} \prod_{(h_P:\prod_{(x:A\sqcup^R B)} P(h(x)))}$
$$\big(H_{\mathsf{set}}(A, h \circ \mathsf{i}, h_P \circ \mathsf{i}) =^P_{\mathsf{setext}'(A,B,R,r,h)} H_{\mathsf{set}}(B, h \circ \mathsf{j}, h_P \circ \mathsf{j})\big)$$

(iii) For all $v : V$, a proof that $P(v)$ is a set (see [17], Lemma 6.9.1)

there exists a function $\varphi : \prod_{(x:V)} P(x)$ such that:

$$\begin{cases} \varphi(\mathsf{set}(A,f)) \equiv H_{\mathsf{set}}(A,f,\varphi \circ f) \\ \mathsf{apd}_{\varphi}(\mathsf{setext}'(A,B,R,r,h)) = H_{\mathsf{setext}'}(A,B,R,r,h,\varphi \circ h) \end{cases}$$

Alternative induction principle The above induction principle is not really suited for proofs. Indeed, it refers to the R-pushout of a bitotal relation R, which we only introduced to be able to properly define V as a higher inductive type. We would like to derive another induction principle, closer to our original specification of definition 2.1. In particular, it should refer to the constructor setext instead of setext′.

The HoTT book [17] gives such an induction principle, but leaves the proof of its equivalence to the former one as an exercise to the reader. When trying to prove it formally in Coq, we have found that the two induction principles actually do not seem equivalent. After discussions with the authors, we defined a weaker induction principle which we actually managed to prove, and which was sufficient to prove all the results in §10.5 of the HoTT book [17]. We discuss this in more details in section 3.2.

2.2 Properties

In this section, we state some of the results that are already known about V. For the proofs of the theorems, see §10.5 of the HoTT book [17].

First, we define a *membership relation* $\in : V \to V \to \mathsf{hProp}$ by induction on the second argument as follows:

$$x \in \mathsf{set}(A,f) :\equiv (\exists (a : A).\, x = f(a))$$

We can also define the *subset relation* $\subseteq$ as usual:

$$x \subseteq y :\equiv \forall (z : V).\, z \in x \Rightarrow z \in y$$

Theorem 2.6. *The following axioms of set theory hold for* $(V, \in)$*:*

(i) extensionality: $\forall (x, y : V).\, x \subseteq y \wedge y \subseteq x \Leftrightarrow x = y$.

(ii) empty set: *for all* $x : V$*, we have* $\neg(x \in \varnothing)$.

(iii) pairing: *for all* $u, v : V$*, there is* $w : V$ *such that* $x \in w \Leftrightarrow x = u \vee x = v$.

(iv) infinity: *there is* $\omega : V$ *such that* $\varnothing \in \omega$ *and* $x \in \omega$ *implies* $x \cup \{x\} \in \omega$.

(v) union: *for all* $v : V$*, there is* $\cup v : V$ *such that* $x \in \cup v \Leftrightarrow \exists (u : V).\, x \in u \in v$.

(vi) function set: *for all* $u, v : V$*, there is* $v^u : V$ *such that* $f \in v^u \Leftrightarrow f$ is a set-theoretic function from u to v.

(vii) $\in$-induction: *if* $C : V \to \mathsf{hProp}$ *is such that* $C(a)$ *holds whenever* $C(x)$ *for all* $x \in a$*, then* $C(v)$ *for all* $v : V$.

(viii) replacement: *given any* $r : V \to V$ *and* $x : V$*, there is* $w : V$ *such that* $y \in w \Leftrightarrow \exists (z : V).\, z \in x \wedge y = r(z)$.

(ix) separation: *given any* $a : V$ *and* $C : V \to \mathsf{hProp}_{\mathcal{U}}$*, there is* $w : V$ *such that* $x \in w \Leftrightarrow x \in a \wedge C(x)$.

The only axiom left to prove in order to get a model of ZF is the powerset axiom. We cannot expect to get powersets directly in HoTT because this would cause impredicativity. However, if we assume the axiom of choice, Diaconescu proved that we also get the principle of excluded middle, and we can then define the powerset of v using functions from v to bool. We get the following theorem:

Theorem 2.7. *In* HoTT *with the axiom of choice, the cumulative hierarchy* V *is a model of Zermelo-Fraenkel set theory with choice,* ZFC.

3 Coq formalization

There are several implementations of HoTT in proof assistants. The oldest ones consist of libraries in already-existing proof assistants, along with some small customizations. The two main ones are the Coq HoTT library that we used, and the Agda library. These libraries have two main flaws: univalence must be explicitly assumed as an axiom, and hence proofs using univalence do not have good computational properties; and the higher inductive types are not supported natively, we must resort to tricks to implement them using regular inductive types.

A first step towards solving the first issue is the recent implementation of a new model of HoTT by M. Bezem and T. Coquand [5]. This new proof assistant, Cubical, allows a computational interpretation of univalence. However, it is still young and experimental, and lacks many of the useful features of Coq and Agda.

On the other hand, B. Barras started implementing a version of Coq which supports higher inductive types. But since the current implementation does not allow the higher inductive types to be recursive, this tool was not sufficient for our purpose.

Our choice of the Coq library was mainly motivated by the fact that I already had some experience with it, and that it already contained most of the general lemmas that we needed. The HoTT library is available at `https://github.com/HoTT/HoTT`. It is based on the "Trunk" version of Coq.

3.1 Higher inductive types in Coq

Currently, no proof assistant fully supports higher inductive types. The Agda HoTT library makes use of the so-called *Private types* to implement them. A custom version of Coq that also supports private types was then implemented by Y. Bertot [4] in order to have higher inductive types in the HoTT library.

The general idea is simple: we define a regular inductive type with the same point constructors as our higher inductive type, then we pose the path constructors as axioms. The induction principle that we get with this definition is stronger than the one we want. Indeed, it lacks the conditions that require the function being defined to be coherent with the path constructors. This is where private types are used: by using the `Private Inductive` notation, we can hide the induction principle so that it cannot be used in the rest of the development. For example, the type of the circle (defined in section 1.8) would be:

```
Private Inductive S1 : Type :=
  | base : S1.
Axiom loop : base = base.
```

We can then define the correct version of the induction principle, and its associated computation rules. Thus, we obtain a type with the desired structure and computational behaviour.

3.2 Results and difficulties

Our Coq development formalizes the results in §10.5 of the HoTT book [17] that deal with constructive set theory (i.e., the last two theorems have been left out because they require the axiom of choice). We also formalized exercise 10.11 of the book, which justifies the definition and induction principle of V (see section 2.1). This work allowed us to find a few mistakes and imprecision in the results given in the HoTT book.

Induction principle. As we already said at the end of section 2.1, we had to modify the induction principle of V in order to make it provable. The one given in the HoTT book is the following, written in pattern-matching language:

Given $P : V \to \mathsf{hSet}$, in order to define $\varphi : \prod_{(x:V)} P(x)$, it suffices to:

(i) For any $A : \mathcal{U}$ and $f : A \to V$, assuming $\varphi(f(a))$ is defined for all $a : A$, construct $\varphi(\mathsf{set}(A, f))$

(ii) For any $f : A \to V$ and $g : B \to V$ satisfying $\mathsf{Eq\_img}(f, g)$, assuming $\varphi(f(a))$ and $\varphi(g(b))$ are defined for all a and b such that $\varphi(f(a)) =^P_p \varphi(g(b))$ whenever $p : f(a) = g(b)$, construct a dependent path $\varphi(\mathsf{set}(A, f)) =^P_{\mathsf{setext}(A,B,f,g)} \varphi(\mathsf{set}(B, g))$

The first clause is just the standard way of dealing with a recursive point constructor such as set, and it didn't cause any problem. However, in the second one, the condition that "$\varphi(f(a)) =^P_p \varphi(g(b))$ whenever $p : f(a) = g(b)$" seemed too strong. To make it provable, we need to state that condition not for *any* p, but for one which comes from the assumption $\mathsf{Eq\_img}(f, g)$. Hence, we replaced the second condition by the following weaker one:

(ii) For any $f : A \to V$ and $g : B \to V$ satisfying $\mathsf{Eq\_img}(f, g)$, assuming $\varphi(f(a))$ and $\varphi(g(b))$ are defined for all a and b such that

$$\begin{aligned} &\big(\forall(a : A).\, \exists(b : B).\, \exists(p : f(a) = g(b)).\, \varphi(f(a)) =^P_p \varphi(g(b))\big) \\ \wedge \quad &\big(\forall(b : B).\, \exists(a : A).\, \exists(p : f(a) = g(b)).\, \varphi(f(a)) =^P_p \varphi(g(b))\big) \end{aligned}$$

construct a dependent path $\varphi(\mathsf{set}(A, f)) =^P_{\mathsf{setext}(A,B,f,g)} \varphi(\mathsf{set}(B, g))$

The proof that this induction principle is indeed provable from the one we gave in section 2.1 can be found in our Coq development under the name `V_rect'`.

Monic set presentation. Most of the difficulties we encountered came from a useful lemma (Lemma 10.5.6 in the HoTT book [17]), which says that any $u : V$ has a canonical representation $u = \mathsf{set}(A_u, m_u)$ where A_u is a type and $m_u : A_u \to V$ is a monomorphism. This result is used later in many of the proofs.

The proof of this lemma requires to be very careful about universe levels. In particular, it uses a bisimulation relation which is a $\mathcal{U}$-small resizing of the equality on V. Even with the recent implementation of universe polymorphism in Coq [15], we had to deal with quite a lot of "Universe Inconsistency" errors. The solution was to specify manually the universe levels in the problematic lemmas, but this was quite tedious since the use of explicit universes is a rather bleeding-edge, and thus poorly-documented feature of Coq.

It is also worth noting that the version of lemma 10.5.6 that we managed to prove is slightly stronger than the one given in the HoTT book: we added the fact that the type A_u that we construct is in fact an h-set. This fact is interesting because it gives us a function of type $V \to \mathsf{hSet}$ (by associating A_u to u), which gives us a link between the sets in V and the h-sets.

Ordered pairs. A technical point was to prove that the ordered pair (a, b), defined as usual as the set $\{\{a\}, \{a, b\}\}$, satisfies the property: $(a, b) = (c, d) \leftrightarrow (a = c) \wedge (b = d)$. This is a bit tedious to prove without excluded middle, I followed Aczel's proof in [2].

4 Open problems

There are two distinct ways of doing set theory in HoTT.

On the one hand, the h-sets of definition 1.6 are spaces whose connected components are contractible. Such a space is homotopically equivalent to a discrete space, i.e., a set. The type hSet of h-sets has been studied extensively in [12]. It is shown to be a model of the so-called structural set theory. From a categorical point of view, hSet forms a ΠW-pretopos.

On the other hand, the type V that we consider in this report is the cumulative hierarchy of sets. It is constructed by starting with the empty set, and iteratively adding new sets containing the previously-constructed ones. In particular, it is well-founded. Although the cumulative hierarchy is a well-known construction in set theory, the definition in HoTT with higher inductive types and mere propositions is new and a number of questions are still open.

4.1 Collection and REA axioms

If we do not assume choice or excluded middle, the axioms of set theory of theorem 2.6 do not allow us to model constructive set theories such as IZF or CZF [2]. Indeed, it is not known whether the *subset collection axiom* or the *(strong) collection axiom* hold. For IZF, we also need the powerset axiom, but this one seems unlikely to hold because it induces impredicativity. Another useful axiom when working in constructive set theories is the *Regular Extension Axiom* (REA) [11], which ensures the existence of inductively defined sets. In this section, we will discuss collection and REA.

Axiom 4.1 (Collection)**.** For every set a,

$$\forall (x \in a).\, \exists y.\, \phi(x, y) \Rightarrow \exists b.\, \forall (x \in a).\, \exists (y \in b).\, \phi(x, y)$$

where $\phi(x, y)$ is a formula with free variables x and y.

The collection axiom is similar to the replacement axiom of theorem 2.6, except that the formula ϕ is not functional: there can be several y associated to the same x. The replacement axiom is provable thanks to unique choice. However, in order to prove collection, it looks like we would need to have choice in HoTT, which is not the case.

We didn't manage to prove that collection doesn't hold in V. One way to do it would be to find a link between the collection axiom in V and the collection axiom in h-sets, which has been studied in [12]. The idea would be to prove that collection in hSet implies collection in V, by finding an appropriate map of type hSet $\to V$. By looking more closely at the proof of lemma 10.5.6 of the HoTT book [17], we have found that the so-called *type of members* gives us a map from V to hSet. However, getting a map in the other direction seems more difficult.

Definition 4.2. A set A is *regular* if it is transitive, inhabited, and for every $a \in A$ and $R \subseteq a \times A$ such that $\forall (x \in a).\, \exists (y \in A).\, [(x, y) \in R]$, there is a set $b \in A$ such that:

$$\forall (x \in a).\, \exists (y \in b).\, [(x, y) \in R] \wedge \forall (y \in b).\, \exists (x \in a).\, [(x, y) \in R]$$

Axiom 4.3 (REA)**.** Every set is a subset of a regular set.

REA seems difficult to prove in HoTT, for similar reasons as collection. We can consider the weaker axiom *functional REA* (fREA) where we ask the relation R to be functional. An interesting sub-question would be to define *hereditary sets* in V:

Definition 4.4. Let A be a set. The class of sets heriditarily an image of a set in A, written $H(A)$, is the smallest class Y such that whenever $b \in A$ and $f : b \to Y$, then $\mathsf{ran}(f) \in Y$. For example, $H(\omega)$ is the set of heriditarily finite sets, i.e., finite sets whose elements are all hereditarily finite sets.

The existence of $H(A)$ for all A follows from $fREA$ [11]. In the next section, we show that $H(\omega)$ is definable in V. We do not know yet for which sets A exactly we can define $H(A)$.

4.2 Construction of $H(\omega)$ in V

First, let us recall the definition of $\omega : V$.

Definition 4.5. Let $\omega : V :\equiv \mathsf{set}(\mathsf{nat}, I)$, where $I : \mathsf{nat} \to V$ is defined recursively on the natural numbers as $I(0) :\equiv \varnothing$ and $I(n+1) :\equiv I(n) \cup \{I(n)\}$.

Informal idea. Given $b \in \omega$ and $f : b \to Y$, the "range of f" kind of looks like what the set constructor gives us: $\mathsf{set}(b, f)$ would be the range of f if the types matched. The problem is that b is a V-set, not a type, so $f : b \to Y$ doesn't make sense. We have to use the so-called type of members $[b]$ defined in the HoTT book [17], whose inhabitants can be mapped to the elements of the set b.

So we would like to define the smallest Y which, for any $b \in \omega$ and $f : [b] \to Y$, contains $\mathsf{set}([b], f)$. This would be something like:

$$Y :\equiv \mathsf{set}\Big(\sum_{b \in \omega} ([b] \to Y), \lambda(b, f).\, \mathsf{set}([b], f)\Big)$$

This means that Y is the set composed of one element $\mathsf{set}([b], f)$ for every $b \in a$ and $f : [b] \to Y$. There are two problems with this definition:

- The types don't match: we are mixing type-theoretic and set-theoretic notations. This is easily fixed: we can note that the set ω is composed of the sets $I(n)$ where $n : \mathsf{nat}$. Hence, instead of $\sum_{(b \in \omega)}$, we can write $\sum_{(n:\mathsf{nat})}$, then replace b by $I(n)$.
- This is impredicative: Y occurs in its own definition, we cannot define it like that.

Formal construction. We're going to use the previous idea, but construct $H(\omega)$ iteratively from a sequence of $(Y_i) : V$.

- $Y_0 :\equiv \varnothing$
- $Y_{i+1} :\equiv \mathsf{set}\big(\sum_{(n:\mathsf{nat})}([I(n)] \to Y_i), \lambda(n,f).\, \mathsf{set}([I(n)], f)\big)$

In fact, there is still one typing issue, here we have $f : [I(n)] \to Y_i$ which doesn't make sense since Y_i is a V-set and not a type. The solution is to have $f : [I(n)] \to V$ along with a proof of $\prod_{(x:[I(n)])}(f(x) \in Y_i)$.

Finally, we can define the function $F : \mathsf{nat} \to V$ such as $F(i) :\equiv Y_i$ by induction on the natural numbers. Then we have $\mathsf{set}(\mathsf{nat}, F)$ which denotes the set $\{Y_0, Y_1, \ldots, Y_i, \ldots\}$. Using the union axiom, we get $H(\omega) :\equiv \bigcup \mathsf{set}(\mathsf{nat}, F)$.

This construction seems roughly generalizable: if we have $a : V$ instead of ω, we can do a similar definition where nat and I are replaced by the type of members $[a]$ and the injection from $[a]$ to the elements of a. It would be interesting to try proving more formally that it works, possibly using the Coq development.

Conclusion

The main purpose of this work was to contribute to the Coq HoTT library by formalizing the theory of chapter 10.5 of the HoTT book [17]. We have successfully proven most of the theorems presented there, and we submitted a pull request to add it to the library.

This formalization was interesting because it allowed us to work with a non-standard higher inductive type. The framework of higher inductive types is still new, and the fact that we had to modify the induction principle of the type V in order to make it provable illustrates how these types are not fully understood yet. Thus, our work allowed us to better understand how some higher inductive types should behave.

Finally, we did some theoretical investigations about the cumulative hierarchy V in a constructive setting. Even though we did not manage to solve any of the big open questions, the definition of hereditary sets is a first step towards a weak version of the regular extension axiom.

Acknowledgements

I would like to thank Robbert Krebbers and Herman Geuvers for our weekly discussions, their advice in writing this report, and Robbert's technical help with Coq. I also thank Mike Shulman, Peter Lumsdane and Jason Gross for discussions.

References

[1] P. Aczel. On relating type theories and set theories. In *Proceedings of TYPES '98, LNCS*. Springer-Verlag, 1998.

[2] P. Aczel and M. Rathjen. Notes on constructive set theory. *Technical Report 40*, 2001.

[3] B. Barras. *Semantical investigations in Intuitionistic Set Theory and Type Theories with inductive families*. Hdr thesis, Université Paris 7, 2012.

[4] Y. Bertot. Private inductive types: Proposing a language extension, 2013, [PDF].

[5] M. Bezem, T. Coquand, and S. Huber. A model of type theory in cubical sets, 2013.

[6] M. Hofmann and T. Streicher. The groupoid interpretation of type theory. In *Venice Festschrift*, pages 83–111. Oxford University Press, 1996.

[7] A. J. C. Hurkens. A simplification of Girard's paradox. volume 902. Springer-Verlag, 1995.

[8] C. Kapulkin, P. L. Lumsdaine, and V. Voevodsky. The simplicial model of univalent foundations, 2012, arXiv:1211.2851.

[9] The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2004, `http://coq.inria.fr`.

[10] A. Miquel. A strongly normalising Curry-Howard correspondence for IZF set theory. *Computer Science Logic*, pages 441–454, 2003.

[11] M. Rathjen and R. S. Lubarsky. On the regular extension axiom and its variants. *Mathematical Logic Quarterly 49*, 2003.

[12] E. Rijke and B. Spitters. Sets in homotopy type theory. 2013, arXiv:1305.3835, special issue "From type theory and homotopy theory to univalent foundations" of MSCS.

[13] A. Setzer. *Proof theoretical strength of Martin-Löf Type Theory with W-types and one universe*. Thesis, University of Munich, 1993.

[14] M. Shulman. A tentative proposal for a general syntax of higher inductive types, 2012, [PDF].

[15] M. Sozeau and N. Tabareau. Universe polymorphism in Coq. *ITP'14*, 2014, [PDF].

[16] T. Streicher. A model of type theory in simplicial sets, 2011, [PDF].

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