Elijah Thompson

Homotopy Type Theory: A Construction of Mathematics

– In Pursuit of Abstract Nonsense –

Thursday 5th May, 2022

Springer Nature

Preface

This text consists of a collection of Homotopy Type Theory notes taken over a summer reading group for HoTT.

Place(s),
month year

Firstname Surname
Firstname Surname

Contents

Part I

Foundations

Chapter 1

Type Theory

1.1 Type Theory versus Set Theory

Homotopy type theory, among other things, is a foundational language for mathematics. We compare and contrast with the set-theoretic Zermelo-Fraenkel construction. The set-theoretic foundation has two “layers": the deductive system of first-order logic, and, formulated inside this system, the axioms of a particular theory, such as ZFC. Thus, set theory is an interplay between sets (the objects of the second layer) and propositions (the objects of the first layer).

By constrast, type theory is its own deductive system: it need not be formulated inside any superstructure, such as first-order logic. Type theory has one basic notion: **types**. Propositions (i.e. statements which we can prove, disprove, assume, negate, and so on) are identified with particular types. The mathematical activity of proving atheorem is thus identified with a special case of the mathematical activity of constructing an object.

This leads to another difference. First, informally a deductive system is a collection of **rules** for deriving things called **judgments**. Thinking of a deductive system as a formal game, the judgments are “position" in the game which we read by following the game rules. We can also think of a deductive system as a kind of algebraic theory, in which case the judgments are the elements and the deductive rules are the operations. From a logical point of view, the judgements can be considered to be the external statements, living in the metatheory, as opposed to the internal statements of the theory itself.

For example, in first-order logic there is only one kind of judgment: that a given proposition has a proof. A rule of first-order logic is a rule of proof construction, which derives judgments from other judgments. The basic judgment of type theory, which in certain cases can be interpreted as “A has a proof", is written “$a : A$” and pronounced as “the term a has type A"m or in HoTT “a is a point of A". When A is a type representing a proposition, then a may be called a **witness** to the provability of A, or **evidence** of the truth of A. In this case the judgment $a : A$ is derivable in type theory for som a precisely when the analogous judgment “A has a proof" is derivable in first order logic.

Note, internally in type theory we cannot “prove" statements about judgments, nor can we “disprove" a judgment. Additionally, we cannot talk about an element “a" in isolation: every element by its very nature is a term of some type, and that type is uniquely determined, generally speaking. That is, “$a : A$" is an **atomic statement** in type theory.

A last major difference between set and type theory is the treatment of equality. The classical notion of equality is a proposition (we can disprove or assume an equality). Since in type theory propositions are types, this means that equality is a type: for elements $a, b : A$, we have a type “$a =_A b$". When $a =_A b$ is inhabited (i.e. has terms), we say that a and b are **(propositionally) equal**.

However, we also need in type theory an equality judgment, existing at the same level as the judgment “$x : A$". This is called **judgmental equality** or **definitional equality**, and we write it as $a \equiv b : A$ or simply $a \equiv b : A$. This can be thought of as meaning “equal by definition". Whether or not two expressions are equal by definition is a matter of expanding out the definitions; in particular, it is algorithmically decidable (though the algorithm is meta-theoretic, not internal to the theory).

If we interpret a deductive system as an algebraic theory, judgmental equality is equality in that theory. A judgmental notion of equality allows us to control the other form of judgment, “$a : A$". Then we should have a rule saying that given the judgments $a : A$ and $A \equiv B$, we may derive the judgment $a : B$. For us type theory will be a deductive system based on two forms of judgment:

- $a : A$ - “a is an object of type A"
- $a \equiv b : A$ - “a and b are definitially equal objects of type A"

As we shall see later, for types A and B we use “$A \longrightarrow B$" as notation for the type of functions from A to B. Then $f : A \longrightarrow B$ can be interpreted as a typing judgment.

Judgments may depend on assumptions of the form $x : A$, where x is a variable and A is a type. For example, assuming A is a type, $x, y : A$, and $p : x =_A y$, we may construct an element $p^{-1} : y =_A x$. The collection of all such assumptions is called the **context**; from a topological point of view this is analogous to a parameter space. Technically the context must be an ordered list of assumptions since we can have later assumptions depend on previous ones.

If the type A in an assumption $x : A$ represents a proposition, then the assumption is a type theoretic version of a hypothesis. Under this meaning of the word assumption, we can assume a propositional equality, by assuming a variable $p : x = y$, but we cannot assume a judgmental equality $x \equiv y : A$, since it is not a type that can have an element. However, if we have a type or an element which involves a variable $x : A$, then we can substitute any particular element $a : A$ for x to obtain a more specific type or element. By the same token we cannot **prove** a judgmental equality either, since it is not a type in which we can exhibit a witness. Nonetheless, we will sometimes state judgmental equalities as part of a theorem, for example “there exists $f : A \longrightarrow B$ such that $f(x) \equiv y$".

1.2 Function Types

Chapter 2

Homotopy Type Theory

Chapter 3

Sets and Logic

Chapter 4

Equivalences

Chapter 5

Induction

Chapter 6

Higher Inductive Types

Chapter 7

Homotopy n-Types

Part II

Mathematics

Chapter 8

Homotopy Theory

Chapter 9

Category Theory

Chapter 10

Set Theory

Chapter 11

Real Numbers

Part III

COQ Proof Assistant

Chapter 12

Coding in Coq

Appendix A

Formal Type Theory

All's well that ends well