Alpha Version

HoTT for Cools

Warm Yourself Up Before Learning Homotopy Type Theory

A Free/Open Source Book, Available At:
www.mehranbaghi.com/hott_for_cools/

Contents

Chapter 1

Preface

Homotopy Type Theory is an exciting and relatively new area of mathematics. The main resource for learning it is a Free/Open Source book with the same name: Homotopy Type Theory: Univalent Foundations of Mathematics. I'm currently reading it and I decided to publish notes that I'm taking from it. For each chapter of the original book we have a corresponding chapter with the same name containing a cheat sheet of theorems of that chapter and some extra notes.

1.1 Disclaimer

- I am not a mathematician
- I am not a native English speaker

So be prepared for lots of mathematical or grammatical mistakes. However, this is a free book, so you can help me make it better. See the How to Contribute section.

- This book is a work in progress and will change frequently:

So always check that you have the latest release.

1.2 How to Contribute

This is a libre/open source project. It is hosted on GitLab and any contribution is welcomed and highly appreciated.

[TODO] Add detailed information on how to commit to a git repository for beginners.

1.2.1 List of Contributors

1.3 Acknowledgment

This book and most of its latex macros is based on the Homotopy Type Theory: Univalent Foundations of Mathematics book.

The PDF, web pages and e-books are produced with scripts around pandoc and other free software. see the GitLab page.

1.4 License

Chapter 2

Introduction

> "Tell readers that Introduction is not essential for book understanding"[1]

The introduction gives a bird's-eye overview of Homotopy type theory and its relation to other fields of mathematics. Unfortunately it intimidates newcomers and they might think that they won't be able to understand the rest of the book. The good news is that the book is almost self sufficient. Although there is one subject that needs more explanation and that is Category theory. As one of the authors says:

> "I think we did a fairly good job of not assuming too much background in topology and in type theory, but we did use a fair number of category-theoretic concepts without really explaining them in much detail"[2]

So it is a good idea to have an introduction to category theory. I don't know much about category theory either, so if you do, and you want to contribute, please let us know by opening an issue on our GitLab.

With that being said, there is a great introductory video series on category theory by Steve Awodey on youtube: Category Theory Foundations, Lecture 1

[1] Andrej Bauer (https://github.com/HoTT/book/issues/727, 2014).
[2] Mike Shulman (https://github.com/HoTT/book/issues/733, 2014).

Chapter 3

Category theory

This chapter contains an introduction to category theory. Category theory is one of the preliminaries to the HoTT book that isn't discussed in much details. In this chapter we will try to give you a super simple foundation of it.

Chapter 4

Type theory

4.1 Type theory versus set theory

Proposition: A statement susceptible to proof.

Theorem: A proposition that has been proven.

Deductive system: A collection of rules for deriving judgments.

Judgment (first order logic): For the proposition A, "A has a proof" is a judgment.

Judgment (type theory): $a : A$ is the basic judgment and pronounced as "the term a has type A".

Propositions are types: Finding an element of a type is proving that proposition. Thus we call an element of a type a *witness* or an *evidence* of the truth of a proposition.

Propositional equality: When $a =_A b$ is inhabited, we say that a and b are propositionally equal.

Judgmental (or definitional) equality: When two expressions are equal by definition. You expand and compute the definitions and conclude that they are equal. We write it as $a \equiv b : A$ or simply $a \equiv b$. Use $:\equiv$ to introduce definitional equality.

Assumption: $x : A$, where x is a variable and A is a type.

Context: The collection of all such assumptions.

Type former: A way to construct types, together with rules for the construction and behavior of elements of that type.

For the rest of this chapter we will introduce these type formers:

- Function types
- Universes and families
- Dependent function types (Π-types)
- Product types
- Dependent pair types (Σ-types)
- Coproduct types
- The type of booleans
- The natural numbers
- Pattern matching and recursion

- Propositions as types
- Identity types
  - Path induction
  - Equivalence of path induction and based path induction
  - Disequality

English	Type Theory
True	$\mathbf{1}$
False	$\mathbf{0}$
A and B	$A \times B$
A or B	$A + B$
If A then B	$A \to B$
A if and only if B	$(A \to B) \times (B \to A)$
Not A	$A \to \mathbf{0}$
For all $x : A$, $P(x)$ holds	$\prod_{(x:A)} P(x)$
There exists $x : A$ such that $P(x)$	$\sum_{(x:A)} P(x)$

How to introduce a new kind of type:

1. **Formation rules:** How to form new types of this kind.
2. **Introduction rules (type's constructors):** How to construct elements of that type by:
   - **Direct definition**
   - **λ-abstraction:** When we don't want to introduce a name for the function.
3. **Elimination rules (type's eliminators):** How to use elements of that type.
4. **Computation rule (β-reduction):** How an eliminator acts on a constructor.
5. **Uniqueness principle (η-expansion) (optional):** Expresses the uniqueness of maps into or out of that type.
6. **Propositional uniqueness principle (optional):** When the uniqueness principle is not taken as a rule of judgmental equality, we can prove it as a *propositional* equality from the other rules for the type.

4.2 Function types

1. **Formation rules:** Given types A and B, we can construct the type $A \to B$ of functions with domain A and codomain B.
2. **Introduction rules:**
   - **Direct definition:** Defining $f : A \to B$ by $f(x) :\equiv \Phi$ where x is a variable and Φ is an expression which may use x.
   - **λ-abstraction:**
   $$(\lambda(x : A).\, \Phi) : A \to B.$$
3. **Elimination rules:** Given a function $f : A \to B$ and an element of the domain $a : A$, we can apply the function to obtain an element of the codomain B, denoted $f(a)$ and called the value of f at a.

4. **Computation rule:** $(\lambda x.\, \Phi)(a) \equiv \Phi'$ where Φ' is the expression Φ in which all occurrences of x have been replaced by a.
5. **Uniqueness principle:** $f \equiv (\lambda x.\, f(x))$. It shows that f is *uniquely* determined by its values.

4.3 Universes and families

Universe: a type whose elements are types.

Hierarchy of universes: $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$ where every universe $\mathcal{U}_i$ is an element of the next universe. We assume that they are cumulative: All the elements of the i^{th} universe are also elements of the $(i+1)^{\text{st}}$ universe.

Typical ambiguity: Omitting the level i for convenience.

Families of types: Using functions $B : A \to \mathcal{U}$ whose codomain is a universe to model a collection of types varying over a given type A.

4.4 Dependent function types

1. **Formation rules:** Given a type $A : \mathcal{U}$ and a family $B : A \to \mathcal{U}$, we may construct the type of dependent functions $\prod_{(x:A)} B(x) : \mathcal{U}$.
2. **Introduction rules:**
   - **Direct definition:** To define $f : \prod_{(x:A)} B(x)$, where f is the name of a dependent function to be defined, we need an expression $\Phi : B(x)$ possibly involving the variable $x : A$, and we write
   $$f(x) :\equiv \Phi \qquad \text{for } x : A.$$
   - **λ-abstraction:**
   $$\lambda x.\, \Phi \;:\; \prod_{x:A} B(x).$$
3. **Elimination rules: Applying** $f : \prod_{(x:A)} B(x)$ to an argument $a : A$ to obtain an element $f(a) : B(a)$.
4. **Computation rule:** Given $a : A$ we have $f(a) \equiv \Phi'$ and $(\lambda x.\, \Phi)(a) \equiv \Phi'$, where Φ' is obtained by replacing all occurrences of x in Φ by a.
5. **Uniqueness principle:** $f \equiv (\lambda x.\, f(x))$ for any $f : \prod_{(x:A)} B(x)$.

Polymorphic function: A class of dependent function types which takes a type as one of its arguments and acts on elements of that type.

4.5 Product types

1. **Formation rules:** Given types $A, B : \mathcal{U}$ we introduce the type $A \times B : \mathcal{U}$, which we call their **cartesian product**.
2. **Introduction rules:**
   - **Direct definition:** Given $a : A$ and $b : B$, we may form $(a, b) : A \times B$.

- **λ-abstraction:**

3. **Elimination rules:** For any $g : A \to B \to C$, we can define a function $f : A \times B \to C$ by

$$f((a, b)) :\equiv g(a)(b).$$

4. **Computation rule:**

$$f((a, b)) :\equiv g(a)(b).$$

5. **Propositional uniqueness principle:** Every element of $A \times B$ is a pair.

Unit type: We call the nullary product type the **unit type** $\mathbf{1} : \mathcal{U}$. The only element of $\mathbf{1}$ is some particular object $\star : \mathbf{1}$.

Chapter 5

Homotopy type theory

> “We seem to get a fair number of people who hit the beginning of chapter 2 and think”oh no, I don’t know what an ∞-groupoid is" and get stuck “. . . ”I would like the message of the book to be that you don’t need to already know what an ∞-groupoid is; you can learn what one is by learning to work with types. But even with that goal, it seems that we have to give some intuitive picture of what this is supposed to mean, and we also have to make connections for the readers who do know what an ∞-groupoid is. Maybe for the 2nd edition we can find some better way of reconciling these goals" Mike Shulman[1]

[1] (https://github.com/HoTT/book/issues/740, 2014).

Chapter 6

To Be Continued…

As I said in the Preface I'm currently reading the original HoTT book and this is a work in progress. So stay tuned and always check for the newest version at https://www.mehranbaghi.com/hott_for_cools/.

Bibliography

Bauer, Andrej. https://github.com/HoTT/book/issues/727, 2014.

Shulman, Mike. https://github.com/HoTT/book/issues/733, 2014.

———. https://github.com/HoTT/book/issues/740, 2014.




These are notes that I'am taking from the original HoTT book.

This is a work in progress. Go to my website to download the latest version or to contribute to the git repository: www.mehranbaghi.com/hott_for_cools/