Introduction to Homotopy Type Theory WiSe19/20

Part I, Lecture 1 Ulrik Buchholtz

Administrative matters

- Course language: English
- Lectures: Mondays 13:30–15:10 in S102/144, Tuesdays 15:20–17:00 in S215/409K
- Office hours: Thursdays 13:30–15:10 in S215/230, or by appointment
- Prerequisites: none (some mathematical maturity and curiosity)
  Optional (but helpful): topology, algebraic topology, algebraic geometry, category theory, functional programming
- Course materials: Egbert Rijke's *Introduction to homotopy type theory*, the *HoTT Book*
- Chat group: `https://hott-intro.zulipchat.com/`
- Course website: `https://www2.mathematik.tu-darmstadt.de/~buchholtz/hott1920/`

Overview and introduction

What is the topic of the course?

$$\text{homotopy type theory} = \text{HoTT} = \text{homotopy (type theory)} = \text{(homotopy type) theory}$$

Here, *type theory*, means formal languages from the foundations of mathematics and computer science that describe *constructions* (e.g., as in Euclidean geometry) and/or *programs*. Specifically, we mean *dependent type theory* in the style of Martin-Löf [5].

And *homotopy types* are a concept that arise in algebraic topology and homotopy theory, capturing objects represented by topological spaces (or simplicial sets). In Part I, we'll use HoTT to understand homotopy types rather than vice versa, so don't worry if you don't know what they are. They also come in variations such as *continuous*, *smooth*, or *equivariant* homotopy types, etc. (This has to do with ∞-toposes a la Lurie [4], but don't worry about that.) In Part II, we'll study homotopy types via simplicial sets and see how they give a model of our type theory.

It was a recent (and perhaps lucky) discovery that connected these very different topics into a unified whole, starting with the *groupoid model* of Hofmann-Streicher [3], but really taking off with Voevodsky's work on the model in simplicial sets (and introduction of the *univalence axiom*) and Awodey-Warren's homotopical interpretation of *identity types* [1].

So why should you care?

- Dependent type theory is a popular foundation for *proof assistants* for computer-checked and -assisted formalization of mathematics: Agda, Coq, Lean, Arend, etc.
- These also allow for *correct-by-construction* development of formally verified computer programs.
- And now they also allow for a *synthetic* account of homotopy theory.
- A small revolution is happening in (some parts of) mathematics, indicating that homotopy types are better foundational building blocks than sets. (For a popular account, see [2].)
- HoTT extends and improves upon dependent type theory, making it better-behaved mathematically (but not quite yet computationally).

- Developing a piece of mathematics in HoTT shows that it has both computational meaning and can be understood continuously/smoothly/in various settings of algebraic geometry. But we can add axioms that are specific to some settings (including classical mathematics!).
- Said in another way: (homotopy) type theory is an *algebraic* account of mathematical constructions.
- It's a vibrant research area with many open problems to work on.

Introduction to homotopy types

Before we start discussing the formal workings of type theory, let us briefly build some intuition for (low-dimensional, truncated) homotopy types. The simplest interesting types are the *propositions*: these are collections (of proofs), and they have the property that any two elements/proofs are equal, or better, are identified. So, classically speaking, a proposition is either empty or a singleton.

The next-simplest types are *sets*: these are collections of elements for which the type of identifications between any two elements is a proposition. Again, speaking classically, any two elements are either equal or they're not, and if they are, there's no interesting information in how they are identified.

So far, so simple. It's at the next level where we depart a little from the usual. A *groupoid* is a collection of elements for which the type of identifications between any two elements is a set. In this case, we also (informally) call the identifications isomorphisms.

Consider for example the type of finite sets. Among its elements we find sets such as $\{a, b, c\}$ and $\{1, 2, 3\}$. And there are 6 different ways of identifying these.

In the usual set-theoretic foundations, we can model groupoids using what we here shall call *set-presented groupoids*[1]. A set-presented groupoid X is given by a set X of objects, and for every pair of elements $x, y \in X$, a set of isomorphisms $X(x, y)$, together with the following structure:

- Identity isomorphisms $1_x \in X(x, x)$ for all $x \in X$.
- Compositions $g \circ f \in X(x, z)$ for all $f \in X(x, y)$ and $g \in X(y, z)$.

All this is required to satisfy the unit and associativity laws, and for every $f \in X(x, y)$ there must be a (necessarily unique) inverse $f^{-1} \in X(y, x)$ with $f^{-1} \circ f = 1_x$ and $f \circ f^{-1} = 1_y$.

A map of set-presented groupoids from X to Y is given by a function $f : X \to Y$ together with functions $f_{x,y} : X(x, y) \to Y(f\,x, f\,y)$ such that $f_{x,x}\,1_x = 1_{f\,x}$ and $f_{x,z}(g \circ f) = f_{y,z}(g) \circ f_{x,y}(f)$.

You may recognize this as a special kind of category. Why don't we take (higher) categories instead of higher groupoids to be the basic objects? One answer is that not all constructions are functorial at the level of categories, for instance the center of a group does not give a functor from the category of groups to that of abelian groups, but it gives a map from the groupoid of groups to that of abelian groups.

We can consider sets as special groupoids, and propositions as special sets, so are groupoids enough to capture all types of mathematical objects? Far from it: Already the type of groupoids is itself a 2-groupoid.

That's it for introduction: Now we'll look at the judgments of type theory, following Egbert's notes, p. 1–7.

Exercises

1. Why is the "center of a group"-construction not functorial with respect to group homomorphisms? (Recall: $Z(G) := \{g \in G \mid \forall_{(h \in G)} gh = hg\}$ with (abelian!) group structure inherited from G.)
2. Exercises 1.1 and 1.2 from Egbert's notes.

[1] Also sometimes called strict groupoids, though they are not themselves groupoids

References

[1] Steve Awodey and Michael A. Warren. "Homotopy theoretic models of identity types". In: *Math. Proc. Cambridge Philos. Soc.* 146.1 (2009), pp. 45–55. ISSN: 0305-0041.

[2] Kevin Hartnett. *With Category Theory, Mathematics Escapes From Equality*. Oct. 19, 2019. URL: https://www.quantamagazine.org/with-category-theory-mathematics-escapes-from-equality-20191010/.

[3] Martin Hofmann and Thomas Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. Oxford Univ. Press, New York, 1998, pp. 83–111.

[4] Jacob Lurie. *Higher topos theory*. Vol. 170. Annals of Mathematics Studies. Princeton, NJ: Princeton University Press, 2009, pp. xviii+925. DOI: 10.1515/9781400830558.

[5] Per Martin-Löf. "Constructive mathematics and computer programming". In: *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*. Ed. by L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski. Vol. 104. Studies in Logic and the Foundations of Mathematics. North-Holland, 1982, pp. 153–175. DOI: 10.1016/S0049-237X(09)70189-2.