Chapter 1

Hott lectures

CMU 2013 Fall: 15-819 Advanced Topics in Programming Languages — Video lectures. Course notes: 15 819: Homotopy type theory

Chapter 2

Hott lecture 1: Introduction

- ITT = intensional type theory.
- ETT = extensional type theory.
- ETT = ITT + ER (equality reflection) + UIP (uniqueness of identity proofs). It's sort of like an intensional theory of sets. Related to Bishop's constructive real analysis.
- HoTT = ITT + HIT (higher inductive types) + UA (univalence axiom). Is an intensional theory of weak infinity groupoids. Types are abstract spaces.

2.1 Brower's program

- Mathematics is a social activity. Thus we need a language for communicating math.
- How is it possible for people to communicate math. ideas? The fundamental idea is that of an algorithm. What all people are capable of doing is to perform a construction.
- Proofs are forms of construction. This has a technical consequence: *Proof relevance*. Proofs are mathmatical objects we can manipulate. A formal proof (ala Godel) is a weak shadow of a proof, since we can enumerate the formal proofs available, but what constitutes a proof is outside of that. Arguably, Godel's incompleteness shows that the set of formal proofs is strictly smaller than the set of proofs.

- HoTT establishes a relationship between proofs of equations v/s paths in a space.
- Homotopy theorists do "assembly on paths". HoTT people perform functional programming on paths.

Synthetic geometry is what Euclid did. Objects are things in themselves where we begin from an axiomatization. Analytic was the Cartesian definition, where a circle is a locus of points, and we start working with real analysis.

Similarly in the case of HoTT, we don't consider paths as $\mathbb{R} \to X$. Rather, we work in Quillen model category / LF / Twelf. the lecturer feels that Coq is more like the traditional approach. HoTT works because we are doing synthetic homotopy theory. This distinction is due to Lawvere.

Type theory is an analysis + categorification of Brouwer's intuitionism drawing on Gentzen's proof theory. A *type* classifies *constructions*. We have *introduction rules* that tell us how to effect the construction, the *elimination rules* tells us how to use a construction. Finally, the inversion rules, which is like a conservation law for proofs, says that elimination after the introduction rule is identity (ie, elim is post-inverse to intro). The introduction/elimination rules are the *statics*, while the inversion rule is the *dynamics*.

Axiomatic freedom of constructive mathematics: Hilbert believed that Brouwer was negating everything that was done so far. However, this is not the case. Fewer assumptions lead to stronger results. Brouwer did not include law of excluded middle.

Computational Trinitariamism: type theory, proof theory, category theory. Starting with proof theory (PT), *Intuistionstic logic* is like "logic as if people matter". Written down in modern form, we have the judgements: `A prop` which meeans that A is a proposition. `A true` which means that A is a true proposition, that is, it *has a proof*. We do not expect that for any proposition, either `A true` or `A false`.

Open endedness: The principle that we do not imagine that we have circumscribed all possible proofs. If a system is not closed ended, we lose, since we can diagonalize our way out of it.

Negative fragment of Intuistionstic propositional logic: Propositions are more than the fact that they are provable, since they have computational content. Thus, we will write down a grammar of proofs. T is the trivially true proposition in what follows.

$$\frac{}{\mathtt{T\ prop}}\ \text{T-formation} \qquad \frac{}{\mathtt{T\ true}}\ \text{T-Introduction}$$

$$\frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \wedge B\ \mathsf{prop}}\ \text{AND-formation}$$

$$\frac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\ \text{AND-introduction}$$

$$\frac{A\ \mathsf{true}\ \wedge\ B\ \mathsf{true}}{A\ \mathsf{true}}\ \text{AND-elimination-left} \qquad \frac{A\ \mathsf{true}\ \wedge\ B\ \mathsf{true}}{B\ \mathsf{true}}\ \text{AND-elimination-right}$$

For entailment, we draw a distinction between entailment ($A \vdash B$) and implication ($A \supset B$) [yes, we write implication as horshoe for reasons that will be explained later apprently]. The idea is something like this: we can consider the collection of monotone maps between posets, which is like considering entailments (since entailments are proof morphisms). It so happens that the collection of monotone maps is itself a poset. This is similar to how we can internalize entailment as the logical statement of implication $A \supset B$.

$$\frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \supset B\ \mathsf{prop}}\ \text{implication-formation}$$

$$\frac{A\ \mathsf{true} \vdash B\ \mathsf{true}}{A \supset B\ \mathsf{true}}\ \text{implication-intro}$$

$$\frac{A \supset B\ \mathsf{true}}{A\ \mathsf{true} \vdash B\ \mathsf{true}}\ \text{implication-elim}$$

We can sometimes uncurry this rule, to get:

$$\frac{A \supset B\ \mathsf{true} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\ \text{implication-elim}$$

Chapter 3

Hott lecture 2: Judgements