Homotopy Type Theory without Homotopy Theory

Thorsten Altenkirch

Functional Programming Laboratory
School of Computer Science
University of Nottingham

January 11, 2013

Type Theory and extensionality

- Type Theory internalizes program extraction.
- If proofs contain programs we need to be able to talk about proofs.
- Extensionality is essential to be able to perform abstractions.
- No need to have separate calculi for concrete and abstract mathematics.
- Indeed, intensional Type Theory, classical set theory and extensional type theory are **not** extensional!
- What is a truly extensional Type Theory?

Homotopy Type Theory

- Homotopy theory: classification of topological spaces by groupoids of paths.
- Observation: Path groupoids correspond to equality types in Type Theory.
- Basic construction in homotopy theory can be modelled by simple constructions in Type Theory.
- Homotopy theory based intuition helps to find proofs in Type Theory.

Goal of this talk

Give an account of the basic concepts of Homotopy Type Theory without any reference to Homotopy Theory.

- Rejection of Uniqueness of Identity Proofs
- Weak equivalence
- Univalence Axiom

Instead we will use the **principle of extensionality**.

Principle of Extensionality

Two objects of the same type should not be both

- indistinguishible (without reference to equality),
- and not provably equal.

- This is a metatheoretic principle not an axiom of type theory.

Functional extensionality

- Consider $f, g : \mathbb{N} \to \mathbb{N}$:

$$\begin{aligned} f\,x &= x + 0 \\ g\,x &= 0 + x \end{aligned}$$

- There is no observation distingushing f and g.
  (without using intensional equality).
- The reason is our black box understanding of functions.
- In Intensional Type Theory there is no proof that $f = g$.
- Hence Intensional Type Theory doesn't satisfy the principle of extensionality for functions.

Functional extensionality

- We can show that

$$\mathrm{congapp} : f = g \to ((n : \mathbb{N}) \to f\ n = g\ n)$$

- We introduce an inverse to $\mathrm{congapp}$:

$$\mathrm{ext} : ((n : \mathbb{N}) \to f\ n = g\ n) \to f = g$$

- Type Theory with ext satisfies the principle of extensionality for functions.

Canonicity

- Adding a constant like ext destroys computational properties of Type Theory.
- E.g. we get closed terms of type $\mathbb{N}$ which contain ext and are not reducible to a numeral.
- This issue can be addressed using the Setoid model, see
  TA. *Extensional equality in intensional type theory.* LICS'99
  TA,C. McBride,W. Swierstra. *Observational equality, now!* PLPV'07
- However, this solution relies on a strong from of proof-irrelevance.

Equality of types

- What is the extensional equality of types?
- Consider A, B : **Type**:

$$A = \mathbb{N}$$
$$B = \mathrm{List}\,1$$

- There is no observation distinguishing A and B.
  (without using intensional equality).
- The reason is that in Type Theory we cannot investigate elements on isolation of their type.
- In Intensional Type Theory (with ext) there is no proof that $A = B$.
- Hence Intensional Type Theory (with ext) doesn't satisfy the principle of extensionality for types.

Equality of types

- We can show that

$$\mathrm{coe}_2 : A = B \to A \simeq B$$

  where $A \simeq B$ means that A is isomorphic to B.
- We introduce an inverse to coe_2:

$$\mathrm{uval}_2 : A \simeq B \to A = B$$

  (univalence for hsets)
- Type Theory with uval_2 satisfies the principle of extensionality for types (actually hsets).
- Indeed, uval_2 implies ext so we get extensionality for functions too.

Uniqueness of identity proofs

- By *uniqueness of identity proofs* (UIP) we mean that any two proofs $p, q : a = b$ should be equal $p = q$.
- UIP is not provable in Intensional Type Theory but it can be proven using pattern matching.
- UIP is inconsistent with uval_2 (just consider the two diferent isomorphisms on Bool).
- *Extensional Type Theory* necessarily features UIP, hence it cannot satisfy the principle of extensionality!

Coherent isomorphism and weak equivalence

- In the absence of UIP we need to refine the notion of isomorphism.
- A function $f : A \to B$ is an isomorphism, iff we have:

$$\begin{aligned} g &: B \to A \\ \alpha &: (a : A) \to g\,(f\,a) = a \\ \beta &: (b : B) \to f\,(g\,b) = b \end{aligned}$$

- This isomorphism is coherent if we additionally have:

$$\Phi : (a : A) \to \mathrm{cong}\,f\,(\alpha\,a) = \beta\,(f\,a)$$

- Equivalently we can require:

$$\Psi : (b : B) \to \mathrm{cong}\,g\,(\beta\,b) = \alpha\,(g\,a)$$

- Coherent isomorphism is isomorphic to weak equivalence (as introduced by Voevodsky)
  This was recently formally verified by Paolo Capriotti in Agda.

General univalence

- Every isomorphism which comes form an equality is coherent.

$$\mathrm{coe} : A = B \to A \approx B$$

  where $A \approx B$ means that A is weakly equivalent (or coherently isomorphic) to B.

- Hence uval_2 is unsound for types which do not satisfy uniqueness of identity proofs. (i.e. are not hsets, e.g. $\mathbf{Set} = \mathbf{Type}_0$).
- Hence it has to be replaced by

$$\mathrm{uval} : A \approx B \to A = B$$

  as an inverse of coe.

- **Conjecture:** Type Theory with uval satisfies the principle of extensionality for types.

Canonicity

- Adding a constant like uval destroys computational properties of Type Theory.
- E.g. we get closed terms of type $\mathbb{N}$ which contain uval and are not reducible to a numeral.
- Our approach using setoids doesn't wotk because we require UIP!
- This is an open problem in Homotopy Type Theory!
- This may be addressed using a semantic interpretation of Homotopy Type Theory
  (e.g. Simplicial Sets or weak ω groupoids).

The role of Homotopy Theory

- Homotopic models (like simplicial sets) show that adding uval is logically sound.
- Homotopy theory provides an excellent intuition and structure for doeng proofs in Type Theory!
- On the other hand we can use Type Theory to formalize proofs in Homotopy Theory elegantly.
- We can also read HTT as Higher-dimensional Type Theory.

Summary

- Homotopy Type Theory seems to satisfy the principle of extensionality.
- Unlike Intensional and Extensional Type Theory.
- We don't know yet how to interpret the univalence axiom computationally.