Towards higher dimensional Type Theory

Thorsten Altenkirch and Thierry Coquand

School of Computer Science
University of Nottingham

April 8, 2011

Background

- New axiom for Type Theory (Univalence)
  inspired by Homotopy Theory
  proposed by Vladimir Voevodsky
- Interesting from a foundational point of view
  new connection between topology and logic.
- Should be explained from a purely type theoretic view
  Connection with representation independence (abstraction)
  relevant for Computer Science
- Voevodsky formalized his approach in Coq
  adapted to Agda and improved by Nisse

Equality of functions

- What should be equality of functions?
- All operations in Type Theory preserve extensional equality of functions.
  The only exception is intensional propositional equality.
- We would like to define propositional equality as extensional equality.

$$
\begin{array}{l}
\mathit{postulate} \\
\quad \mathit{ext} : (f\ g : A \to B) \\
\quad\quad \to ((a : A) \to f\ a \equiv g\ a) \to f \equiv g
\end{array}
$$

Equality of types

- What should be equality of types?
- All operations of Type Theory preserve isomorphisms (or bijections).
  The only exception is intensional propositional equality.
- Unlike Set Theory, e.g. $\{0, 1\} \simeq \{1, 2\}$ but $\{0, 1\} \cup \{0, 1\} \not\simeq \{0, 1\} \cup \{1, 2\}$.
- We would like to define propositional equality of types as isomorphism.

UIP and isomorphism

- Uniqueness of identity proofs (UIP)

$$uip : (a\ b : A)\ (p\ q : a \equiv b) \rightarrow p \equiv q$$

- UIP doesn't hold if we define equality of types as isomorphism.
- E.g. there is more than one way to prove that *Bool* is isomorphic to *Bool*.
- If we want to use isomorphism as equality we cannot allow uip.
- In Agda that can be achieved by using the new flag $-noK$ (experimental).

Dimensions of types (or h-levels)

- A type is contractible if it contains precisely one element.

$$Contr\ A = \Sigma\ [a : A]\ ((a' : A) \to a \equiv a')$$

- Contractible types have dimension 0.
- A type has dimension $n + 1$, if its equality is n-dimensional.
- The 1-dimensional types are the *propositions* (any two proofs are equal).
- The 2-dimensional types are the *sets* (their equality is propositional).
- The universe of small sets with isomorphism as equality is 3-dimensional.

Some results

- Contractibility *Contr A* is of dimension 1 (propositional).
- Similar, the predicate *Dim n A* (being n-dimensional) is also of dimension 1 (propositional).
- $A \to \mathit{Contr}\ A$ is equivalent to A being propositional.
- The product of contractible types is contractible.

  $$((x : A) \to \mathit{Contr}\ (B\ x)) \to \mathit{Contr}\ ((x : A) \to B\ x)$$

  This is equivalent to functional extensionality.
- In general all dimensions are closed under Π-types.

From bijection to weak equivalence

- A function $f : A \to B$ is a bijection if there is precisely one inverse for any $b : B$.

$$\mathit{bijective}\ f = (b : B) \to \exists ! [a : A]\ f\ a \equiv b$$

- *bijective f* is only a proposition, if B is a set.
- We can fix this by demanding that the equality proof is unique too:

$$\mathit{isWeakEquivalence}\ f = (b : B) \to \mathit{Contr}\ (\Sigma\ [a : A]\ f\ a \equiv b)$$

- Can be rewritten as:

$$\mathit{isWeakEquivalence}\ f = (b : B) \to \mathit{Contr}\ (f^{\ -1}\ b)$$

using

$$\_^{\ -1} : (f : A \to B)\ (b : B) \to \mathit{Set}$$
$$(f^{\ -1})\ b = \Sigma\ [a : A]\ (f\ a \equiv b)$$

Univalence

- Two types are weakly equivalent $A \approx B$ if there exists a weak equivalence between them.
- The Univalence axiom states that equality of sets is weak equivalence.
- Weak equivalence $A \approx B$ is logically equivalent to isomorphism.
- But it isn't weakly equivalent to isomorphism (or isomorphic to it).
- Weak equivalence (isomorphism) is stronger than logical equivalence.
- Surprisingly: Univalence implies functional extensionality (*ext*).
- Isomorphic structures are equal (shown for one simple example by Thierry and Nisse).

Type Theory with Univalence

- We can add Univalence as a postulate.
- This destroys canonicity
  (e.g. there are non-standard natural numbers).
- Can we justify the univalence axiom constructively? I.e. can we give computation rules?
- This is similar to the problem of elimination of functional extensionality.
- Idea: Exploit that all operations are closed under functional extensionality and isomorphisms.
- Additional complexity: we cannot assume UIP.
- Also interpret (proof-relevant) quotients.

Summary

- Type Theory without UIP
  offers a more abstract view on sets and structures
  reflects mathematical practice
  (avoid dependendence on representation choices)
  Also relevant for Computer Science
- New ways of understanding this theory
  comes from homotopy.
  Does this help us?
- Not clear yet how to give a computational interpretation
  continuing work on elimination of extensionality
  but it is much harder.