Type Theory and Formalization of Mathematics

Anders Mörtberg

Institute for Advanced Study

September 28, 2015

Introduction

I work on computer formalization of mathematics, this means that I implement mathematical proofs in a proof assistant

Working in a proof assistant is a bit like writing proofs in LaTeX, but the system also checks that the proofs are correct

Examples of proof assistants: **Coq**, Agda, Isabelle, HOL, Mizar...

Automated vs. Interactive theorem proving

There are two different approaches to theorem proving on computers:

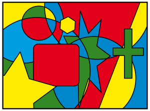
- **Automated** theorem proving: the user provides statements and the computer tries to **find** proofs (usually not decidable)
- **Interactive** theorem proving: the user provides proofs and the computer **checks** if the proofs are correct (usually decidable)

In this talk I will only talk about interactive theorem proving

Famous computer formalizations

Recently many impressive formalizations have been performed:

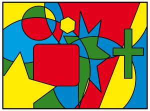
- 2005: The four color theorem in Coq (Gonthier *et al.*):



Famous computer formalizations

Recently many impressive formalizations have been performed:

- 2005: The four color theorem in Coq (Gonthier *et al.*):



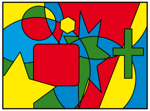
- 2012: The Feit-Thompson theorem in Coq (Gonthier *et al.*):

```
Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :
  odd #|G| -> solvable G.
```

Famous computer formalizations

Recently many impressive formalizations have been performed:

- 2005: The four color theorem in Coq (Gonthier *et al.*):



- 2012: The Feit-Thompson theorem in Coq (Gonthier *et al.*):

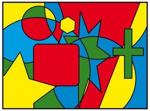
```
Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :
  odd #|G| -> solvable G.
```

- 2014: The Kepler conjecture in Isabelle/HOL (Hales *et al.*):

Famous computer formalizations

Recently many impressive formalizations have been performed:

- 2005: The four color theorem in Coq (Gonthier *et al.*):



- 2012: The Feit-Thompson theorem in Coq (Gonthier *et al.*):

```
Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :
  odd #|G| -> solvable G.
```

- 2014: The Kepler conjecture in Isabelle/HOL (Hales *et al.*):

All of these systems are based on **type theories**

Type theory

Type theory considers a class of formal systems which can be seen as an alternative to set theory for the foundations of mathematics

1908 Russell: Mathematical Logic as Based on the Theory of Types
(1908 Zermelo: Untersuchungen über die Grundlagen der Mengenlehre)

1940 Church: A Formulation of the Simple Theory of Types

Many modern type theories and proof assistants (*e.g.* Coq) are based on work by Per Martin-Löf from the 1970s (I will refer to these as "MLTT")

Martin-Löf Type Theory: basic judgments

Basic judgment forms:

- A is a type
- a is an element of a type A
- A and B are equal types
- a and b are equal elements of type A

MLTT has function types, product types, sum types...

Martin-Löf Type Theory: examples

Examples:

$$\begin{aligned} \mathbb{N} &\text{ is a type} \\ 2 &: \mathbb{N} \\ + &: \mathbb{N} \to \mathbb{N} \to \mathbb{N} \\ (2,4) &: \mathbb{N} \times \mathbb{N} \end{aligned}$$

One difference with set theory is that each term only have one type

MLTT and the Brouwer-Heyting-Kolmogorov interpretation

Proofs are “first-class citizens”: an even natural number is a pair of the number and a proof that it is even

MLTT and the Brouwer-Heyting-Kolmogorov interpretation

Proofs are "first-class citizens": an even natural number is a pair of the number and a proof that it is even

This gives an interpretation of the logical connectives. For instance an implication:

$$P \to Q$$

is a **function** that maps a proof of P to a proof of Q.

MLTT and the Brouwer-Heyting-Kolmogorov interpretation

Proofs are "first-class citizens": an even natural number is a pair of the number and a proof that it is even

This gives an interpretation of the logical connectives. For instance an implication:

$$P \to Q$$

is a **function** that maps a proof of P to a proof of Q.

Propositions form a "sub-universe" of the types and logical connectives are encoded by the general operations on types

Martin-Löf Type Theory: syntax

MLTT uses the syntax of λ-calculus, just like functional programming languages (like Lisp, Scheme, Haskell, OCaml...), hence it can be seen as a functional programming language

Because of this it is well suited as a system for computer formalization

Univalent Foundations

Univalent Foundations aims at providing a **practical** foundations of mathematics built on top of MLTT

Started by Vladimir Voevodsky around 2006–2009

The IAS had a special year devoted to it 2012–2013

Is being actively developed in Coq in the UniMath library

The Univalence axiom

Univalent foundations extends MLTT with:

Univalence axiom: An equivalence of types lifts to an equivalence of all constructions on those types

The Univalence axiom

Univalent foundations extends MLTT with:

Univalence axiom: An equivalence of types lifts to an equivalence of all constructions on those types

One consequence of this axiom is that it is possible to **transport** structures along equivalences (cf. Bourbaki: Theory of sets, 1968)

Transporting structures along equivalences

One can represent $\mathbb{C}$ either using Cartesian or polar coordinates. These two representations are equivalent.

Transporting structures along equivalences

One can represent $\mathbb{C}$ either using Cartesian or polar coordinates. These two representations are equivalent.

If we prove that the Cartesian representation is a field we can **transport** this structure along the equivalence to get a field structure on the polar representation.

Transporting structures along equivalences

One can represent $\mathbb{C}$ either using Cartesian or polar coordinates. These two representations are equivalent.

If we prove that the Cartesian representation is a field we can **transport** this structure along the equivalence to get a field structure on the polar representation.

Univalence is a general formulation of this transport property in MLTT

Transporting structures along equivalences

Open problem: Find an algorithm for transporting structures along equivalences in MLTT

This is not only for equivalence of types, but for equivalences of general mathematical structures: isomorphisms of groups, equivalences of categories, etc.

Transporting structures along equivalences

A model of MLTT is a category with some additional structure

Transporting structures along equivalences

A model of MLTT is a category with some additional structure

Theorem: MLTT with the Univalence axiom has a model in Kan simplicial sets (Voevodsky)

Transporting structures along equivalences

A model of MLTT is a category with some additional structure

Theorem: MLTT with the Univalence axiom has a model in Kan simplicial sets (Voevodsky)

Problem: This model uses classical logic in essential ways (Bezem, Coquand)

Transporting structures along equivalences

A model of MLTT is a category with some additional structure

Theorem: MLTT with the Univalence axiom has a model in Kan simplicial sets (Voevodsky)

Problem: This model uses classical logic in essential ways (Bezem, Coquand)

Goal: Find a constructive model and extract an algorithm

Transporting structures along equivalences

A constructive model was recently presented in:

Bezem, Coquand, Huber: A Model of Type Theory in Cubical Sets, 2014

Transporting structures along equivalences

A constructive model was recently presented in:

Bezem, Coquand, Huber: A Model of Type Theory in Cubical Sets, 2014

Work in progress (with Coquand, Cohen, Huber): construct a simpler(?) constructive model based on cubical sets with connections

Transporting structures along equivalences

A constructive model was recently presented in:

Bezem, Coquand, Huber: A Model of Type Theory in Cubical Sets, 2014

Work in progress (with Coquand, Cohen, Huber): construct a simpler(?) constructive model based on cubical sets with connections

We have used this to build a type theory with a computational interpretation of Univalence; in particular we have an algorithm for transporting structures along equivalences in this type theory:

`https://github.com/mortberg/cubicaltt`

Thank you for your attention!