Applications of Type Theory to Univalent Mathematics

Thierry Coquand

Autumn School, Proof and Computations, September 2019

What I will try to present

(1) Motivations:

-what is univalent mathematics

-why type theory?

(2) Basic notions: homotopy level, univalence and propositional truncation

(3) Structure Identity Principle

(4) Group theory in univalent mathematics and cohomology of types

Type theory

de Bruijn *The mathematical language AUTOMATH*
Symposium of Automatic Demonstration, IRIA, Versailles, Dec. 1968

Howard *The formulae-as-types notion of construction.* Manuscript 1969.

de Bruijn's papers are available at https://www.win.tue.nl/automath/

In particular

N.G. de Bruijn, *AUTOMATH, a language for mathematics*, 1973

L.S. van Benthem Jutting, *The development of a text in AUT-QE*, 1973

AUTOMATH

Treating propositions as types is definitely not in the way of thinking of ordinary mathematician, yet it is very close to what he actually does

A survey of the project AUTOMATH
N.G. de Bruijn, 1980, in
To H.B. Curry, essays on combinary logic, lambda-calculus and formalism

AUTOMATH

Representation of statements like

Theorem 1: *Let x be a real number such that $f(x) > 1$ and let n be a natural number. If we have $g(x) > x^n$ then $f(x) > n$.*

When a mathematician wants to use this statement, with $x = \pi$ and $n = 5$ he has to give a proof (1) of $f(\pi) > 1$ and then a proof (2) of $g(p) > \pi^5$

He can show the statement $f(\pi) > 5$ by reference to theorem 1 and by giving *in this order*

π, the proof (1), 5 and the proof (2)

AUTOMATH

In AUTOMATH this can be represented as

corollary $=$ Theorem 1(π, (1), 5, (2)) : A

where A is the statement $f(\pi) > 5$

AUTOMATH

One crucial notion inspired by the notion of *block structure* in ALGOL 60 is the notion of *context*: sequence of declarations of object variables with their types and names of hypotheses with the corresponding statement and this in an *arbitrary* order

$x : R,\ h_1 : f(x) > 1,\ n : N,\ h_2 : g(x) > x^n$

AUTOMATH had a primitive “sort” type

$R : \text{type},\ N : \text{type},\ x : R,\ h_1 : f(x) > 1,\ n : N,\ h_2 : g(x) > x^n$

One could also introduce a primitive sort prop or to have prop = type

AUTOMATH

AUTOMATH used the same notation for typed abstraction and implication introduction and forall introduction

The basic notions are

-context (base space)

-type in a given context (fibration)

-element of a type in a given context (section)

AUTOMATH

A crucial point is that $a : A\ (\Gamma)$ can be *decided* by a simple uniform procedure

Represents well what a mathematician is doing when he is checking a proof

AUTOMATH

Distinction between two kind of equalities

-definitional equality: terms are equal by definition

This is the equality used in type-checking

-mathematical equality: which is represented as a type

The definitional equality belongs to the metalanguage

AUTOMATH

Uniform treatment of *propositions* and *types*

(A type is a collection of mathematical objects)

E.g. modus ponens is represented by function application

This plays a crucial role in the representation of univalent mathematics

Univalent mathematics?

See the motivations of Voevodsky

In his home page: Notes on Homotopy Lambda Calculus, 2006

Talk 2006 on homotopy theory and foundations of mathematics

Situation reminiscent of 19th century (what is a continuous function?)

Problem with refereeing; how to be sure of the correctness of a proof?

Univalent mathematics?

Because of the amazing intuitiveness of the basic concepts of category theory this foundational insufficiency is not (yet) a serious problem in our everyday work. In my experience it becomes noticeable only at the level of 2-categories which may be the reason why 2-categorical arguments are not very common in mathematics. The real need of formalized foundations arises when one attempts to use a computer to verify a proof.

Univalent mathematics?

It is a fact of life that proofs in contemporary mathematics are getting too long, complicated and numerous to be rigorously checked by the community. Moreover it is precisely the most technical parts of the proofs where the probability of a mistake is the highest which are most likely to be skipped in the verification process. Since the ability to build on layers and layers of earlier results is probably the most important feature which underlies the success of mathematics as an enterprise this is a very serious problem and without finding its solution mathematics can not move forward.

Univalent mathematics?

Generalization of sheaves

“Sheaves” of structures: we need “groupoid-valued” sheaves

Notion of *stacks*, algebraic geometry

Then we need 2-stacks

See also the preface of Lurie *Higher topos theory*

It gives a suggestive motivation for introducing these “higher” objects

What is a ∞-topos?

Univalent mathematics?

How to deal with this?

André Joyal was suggesting to use the notion of *quasi-category*

This was introduced by Boardman and Vogt 1973

Based on simplicial sets, highly non effective

Weakening of the Kan filling conditions

A "∞-groupoid" will be represented by a Kan simplicial set

Univalent mathematics?

E.g. notion of initial object: the space of arrows from the initial object should be *contractible*

The space of initial objects is itself contractible if inhabited

Takes one page to prove

Lurie *Higher Topos Theory*, more than 900 pages just to present the basic definitions and properties of higher order topos

Univalent mathematics?

Voevodsky 2006, talk at the IAS

The language of *dependent types* seems perfect to express these higher notions

This is what I want to explain in these lectures

A spartan (intensional) Martin-Löf type theory

1. Base types $\mathbb{0}$, $\mathbb{1}$, $\mathbb{2}$, $\mathbb{N}$.
2. Cartesian products $A \times B$ and $\Pi(x : X), A(x)$.
3. Function types $X \to A$ also written A^X.
4. Sums $A + B$ and $\Sigma(x : X), A(x)$.
5. A large type $\mathcal{U}$ of small types (universe).

   Sometimes an extra large type $\mathcal{V}$ of large types.
6. An identity type $\mathrm{Id}_X(x, y)$ for any $X : \mathcal{U}$ and $x, y : X$.

Curry–Howard propositional logic

Given two propositions (that is, types) A and B, we define

1. Conjunction. $A \wedge B \equiv A \times B$.

   "A proof of $A \wedge B$ is a pair (a, b) consisting of a proof a of A and a proof b of B."

2. Disjunction. $A \vee B \equiv A + B$.

   "A proof of $A \vee B$ is either a proof of A or a proof of B."

3. Implication. $A \to B$ is the function space, which has the same notation.

   "A proof of $A \to B$ transforms any proof of A into a proof of B."

4. Negation $\neg A \equiv A \to \mathbb{0}$.

Curry–Howard quantifiers

Given a family A of propositions (that is, types) indexed by a type X, we have:

1. Universal quantification: $\forall(x : X), A(x) \equiv \Pi(x : X), A(x)$.

   "A proof of $\forall(x : X), A(x)$ is a function that gives a proof of $A(x)$ for any given $x : X$."

2. Existential quantification: $\exists(x : X), A(x) \equiv \Sigma(x : X), A(x)$.

   "A proof of $\exists(x : X), A(x)$ is a pair (x, a) consisting of a witness $x : X$ and a proof a of $A(x)$."

Curry–Howard logic

false	$\bot$	$\mathbb{0}$	empty type
true	$\top$		any type for which a point can be exhibited
and	$\wedge$	$\times$	cartesian product of two types
or	$\vee$	$+$	disjoint sum of two types
implies	$\to$	$\to$	function space
for all	$\forall$	Π	product of a type family
for some	$\exists$	Σ	disjoint sum of a type family
equals	$=$	Id	identity type

Martin-Löf introduced the identity type precisely to extend Curry-Howard logic with equality.

Gentzen’s natural deduction and dependent types

Each connective defined by introduction rule(s)

Associated elimination principle

This was revisited by Martin-Löf with dependent type

E.g. elimination for $+$ or for identity

Closely connected to the notions of *data types* and *constructors*

Prawitz, Martin-Löf 1960s

McCarthy, Landin 1960s

Example: there are infinitely many prime numbers

$$f : \Pi(n : \mathbb{N}), \Sigma(p : \mathbb{N}), (p > n) \times \text{isPrime}(p).$$

1. A point of this type is a function f that for any $n : \mathbb{N}$ gives a prime $p > n$.
2. The type $n > p$ can be defined by induction on n and p like we defined the identity type $m = n$.

   Or in many other equivalent ways, e.g. $(n > p) \equiv \Sigma(k : \mathbb{N}), p + k + 1 = n$, after we have defined addition by induction.
3. After we define multiplication by induction, we define isPrime(p) in the usual way.

A lot of mathematics can be formulated and proved in this way.

Preparation: Univalence gives function extensionality (funext)

Pointwise equal functions are equal:

$$\Pi(X:\mathcal{U}),\Pi(A:X\to\mathcal{U}),\Pi(f,g:\Pi(x:X),A(x)),(\Pi(x:X),f(x)=g(x))\to f=g.$$

- Something we don't have in intensional Martin-Löf type theories.
- We don't have univalence yet, so we assume funext when needed, for the moment.

Univalent propositions

Or *propositions*, for short, also known as *h-propositions* or *mere propositions*.

(But “mere” is too derrogatory for my taste, and “h-” is strange.)

1. A type X is a proposition if any two of its elements are equal:

   $$\text{isProp}(X) \equiv \Pi(x, y : X), x = y.$$

   We also say that X is a subsingleton.

2. We have funext $\to$ isProp(isProp(X)).

   Being a proposition is itself a proposition.

3. Assuming *funext*, also any two maps into the same proposition are equal.

Example of a non-trivial type that is a univalent proposition

$$\Sigma(n : \mathbb{N}), \mathrm{isPrime}(n) \times (\text{n is the difference of two squares of primes}).$$

1. Although propositions are subsingletons, they are not necessarily "proof-irrelevant".
   - They have information content.
   - The number 5 can be extracted from any proof of this proposition.
2. But for the above type to really be a proposition, we additionally need:
   - propositional extensionality (any two logically equivalent propositions are equal).
   - Which is not provable in MLTT, but follows from univalence, like funext.

A subtler example (the mystery of MLTT's identity type)

Although the type $x = y$ need not be a subsingleton for $x, y : X$, the type

$$\Sigma(x : X), x = y$$

is always a subsingleton for any $y : Y$, and in fact even a singleton (or contractible):

$$\mathrm{isSingleton}(A) \equiv \Sigma(a_0 : A), \Pi(a : A), a = a_0.$$

This doesn't require propositional or function extensionality, or anything beyond MLTT.

Theorem of MLTT. $\Pi(y : X), \mathrm{isSingleton}(\Sigma(x : X), x = y)$.

Non-Theorem of MLTT. ~~$\Pi(x, y : X), \mathrm{isSubsingleton}(x = y)$~~.

An even subtler, crucial example (Voevodsky)

Let $f : X \to Y$ be a function of two types X and Y.

1. The type

   $$\mathrm{isIsomorphism}(f) \equiv \Sigma(g : Y \to X), (g \circ f = \mathrm{id}_X) \times (f \circ g = \mathrm{id}_Y)$$

   need not be a proposition.

   Because of the potential presence, in MLTT, of higher-dimensional types.

2. However, the type

   $$\mathrm{isEquivalence}(f) \equiv \Pi(y : Y), \mathrm{isSingleton}(\Sigma(x : X), f(x) = y)$$

   always is a proposition, assuming functional extensionality.

   (The latter is a retract of the former.)