Univalent Type Theory

Thierry Coquand

Tutorial for the Logic Colloquium 2016, Leeds

Foundation of mathematics

《nowadays it is known to be possible, logically speaking, to derive practically the whole of known mathematics from a single source, the Theory of Sets . . . By so doing we do not claim to legislate for all time. It may happen at some future date that mathematicians will agree to use modes of reasoning which cannot be formalized in the language described here; *according to some, the recent evolution of axiomatic homology theory would be a sign that this date is not so far*. It would then be necessary, if not to change the language completely, at least to enlarge its rules of syntax. But this is for the future to decide.》

Bourbaki, Introduction of Theory of Sets

Description of mathematical objects

Basis: algebraic structures, ordered structures

E.g. groups, rings, lattices

Set equipped with some operations and/or relations satisfying some properties

Two isomorphic structures can be considered to be the 《same》

At this level one can talk about 《initial structures》

Uniqueness up to isomorphism

This is the level considered by Bourbaki in his *théorie des structures*

Description of mathematical objects

Two isomorphic groups G and H satisfy the same $\langle\!\langle$structural $\rangle\!\rangle$ properties

If G is abelian, so is H

If G is solvable, so is H

But we can have $\pi \in G$ and $\pi \notin H$

Transportable properties (Bourbaki)

$\langle\!\langle$When two relations have the same structure, their logical properties are identical, except such as depend upon the membership of their fields$\rangle\!\rangle$

Russell (1959) *My philosophical development*

Description of mathematical objects

Next level: collection of all groups, or all sets (at a fixed universe)

When are two such collections considered to be the «same»?

Let B be a given set

For a mathematician the two collections SET^B and SET/B are «identical»

They satisfy the same «structural»/transportables properties

SET^B contains families of sets $X_b,\ b \in B$

SET/B contains $Y, f \in B^Y$, functions of codomain B

Description of mathematical objects

We have two canonical maps $F : \mathsf{SET}^B \to \mathsf{SET}/B$ and $G : \mathsf{SET}/B \to \mathsf{SET}^B$

$F(X) = (b : B) \times X_b, \pi_1$

$G(Y, f) = (f^{-1}(b))_{b \in X}$

$G(F(X))$ and X are only isomorphic (and not equal as sets in general)

$G(F(X))_b = \{b\} \times X_b$

The two collections (groupoids) SET^B and SET/B are *equivalent*

F and G do *not* define an isomorphism

Collapsing

We get a *new* way to identify collections

The collection of all linear orders with 27 elements is a large collection

In set theory, it forms a class and not a set

But it is 《the same》 as the groupoid with one object and one morphism

Description of mathematical objects

This natural stratification of collection of mathematical objects is not directly represented in set theory

A (Grothendieck) universe is a set as any other set with its notion of equality

Similarly for a collection of structures

Bourbaki has a characterisation of transportable properties for his notion of structure

Such a description for the next level is much more complex

It should be impossible to formulate a statement which is not invariant with respect to equivalences

Description of mathematical objects

At the next level we have 2-*groupoids*

Then n-groupoids, then ∞-groupoids

More and more complex notions of equivalences

Less and less clear when a property is transportable along equivalences

The problem is to describe formally what are the laws of these equivalences

What is surprising is that it can be done in an uniform way

Two *new* laws for equality (Martin-Löf 1973, Voevodsky 2010)

Set theory and type theory

1908 Zermelo *Untersuchungen über die Grundlagen der Mengenlehre*

1908 Russell *Mathematical Logic as Based on the Theory of Types*

«Simple» type theory

1940 Church *A Formulation of the Simple Theory of Types*

Extremely simple and natural

A type $bool$ as a type of «propositions»

A type I for «individuals»

Function type $A \to B$ and we can add product type $A \times B$

For example the identity function is of type $A \to A$

Natural semantics of *types* as *sets*

Functions in simple type theory

In set theory, a function is a *functional graph*

In (this version of) type theory, a function is given by an *explicit definition*

If $t : B$, we can introduce f of type $A \to B$ by the definition

$$f\ x = t$$

$f\ a$ «reduces» to $(a/x)t$ if a is of type A

We use the notation $f\ a$ for $f(a)$, which comes from combinatory logic

Functions in simple type theory

For instance, we can define $id : A \to A$ by

$id\ x = x$

If $f : A \to B$ and $g : B \to C$ we can define

$g \circ f \quad : \quad A \to C$

$(g \circ f)\ x = g\ (f\ \ x)$

Functions in simple type theory

We have two notions of function

-*functional graph*

-*function explicitly defined* by a term

What is the connection between these two notions?

Church introduces a special operation $\iota x.P(x)$ and the ≪axiom of description≫

If $\exists ! x : A.P(x)$ then $P(\iota x.P(x))$

Functions in simple type theory

We can then define a function from a functional graph

$\forall x.\exists ! y.R(x,y) \rightarrow \exists f.\forall x.R(x,f(x))$

by taking $f\ x \ = \ \iota y.R(x,y)$

By contrast, Hilbert's operation $\epsilon x.P(x)$ (also used by Bourbaki) satisfies

if $\exists x : A.P(x)$ then $P(\epsilon x.P(x))$

To use $\exists ! x : A.\varphi$ presupposes a notion of equality on the type A

Rules of equality

Equality can be specified by the following purely logical rules

(1) $a =_A a$

(2) if $a_0 =_A a_1$ and $P(a_0)$ then $P(a_1)$

Given (2), the law (1) is equivalent to the fact that $a_0 =_A a_1$ is implied by

$\forall P.\ P(a_0) \rightarrow P(a_1)$

Equality in mathematics

The *first* axiom of set theory is the axiom of *extensionality* stating that two sets are equal if they have the same element

In Church's system we have two form of the axiom of extensionality

(1) two equivalent propositions are equal

$$(P \equiv Q) \;\rightarrow\; P =_{bool} Q$$

(2) two pointwise equal functions are equal

$$(\forall x : A.f\ x =_B g\ x) \quad \rightarrow \quad f =_{A \rightarrow B} g$$

The axiom of univalence will be a generalization of (1)

Simple type theory

⟪ The simple theory of types provides a straightforward ... foundation for the greater part of classical mathematics. That is why a number of authors (Carnap, Gödel, Tarski, Church, Turing) gave a precise formulation of it, and used it as a basis for metamathematical investigations. The theory is straightforward because it embodies two principles which (at least before the advent of modern abstract concepts) were part of the mathematicians normal code of practice. Namely that a variable always has a precisely delimited range, and that a distinction must always be made between a function and its arguments. In this sense one might claim that all good mathematicians had anticipated simple type theory. [Indeed Turing made this claim for primitive man]. ⟫

Robin Gandy *The simple theory of types*, Logic Colloquium, 1976

Type theory and set theory

In his 1931 paper

On Formally Undecidable Propositions of Principia Mathematica and Related Systems I

Gödel uses a restriction of this system with only the types

$1 = I$

$2 = I \to bool$

$3 = (I \to bool) \to bool$

$\ldots$

Type theory and set theory

The *atomic formula* are of the form $b\ u$ where b of type $n+1$ and u of type n

This can be read as: u belongs to the class b

If we start from I empty collection, we get a cumulative hierarchy

Type theory and set theory

Extensionality axiom

$$(\forall x_n\ (a\ x_n \leftrightarrow b\ x_n))\ \rightarrow\ a =_{n+1} b$$

《A class is completely determined by its elements》

Neither pairing, nor function type

-pairs can be defined (N. Wiener 1914, Kuratowski) 《variables for binary or n-ary functions (relations) are superflous as basic signs》

-functions can be defined as functional graphs

Type theory and set theory

In set theory, we usually start with I empty collection

Transfinite iteration and cumulativity

$\langle\langle$Types$\rangle\rangle$ become ordinals

We can quantify over variable ranging over all $\langle\langle$types$\rangle\rangle$

An infinite type of individuals is obtained at stage ω

Type theory and set theory

《It is a pity that a system such as Zermelo-Fraenkel set theory is usually presented in a purely formal way, because the *conception* behind it is quite straightforwardly based on type theory. One has the concept of an arbitrary *subset of a given domain* and that the collection of *all subsets* of the given domain can form a new domain (of the next type!). Starting with a domain of individuals (possibly empty), this process of forming subsets is then iterated into the *transfinite*. Thus, each set has a type (or *rank*), given by the ordinal number of the stage at which it is first to be found in the iteration.》

Dana Scott, *A type-theoretical alternative to ISWIM, CUCH, OWHY*, 1969

Universes

Simple type theory is elegant but presents unnatural limitations

We cannot express the notion of 《arbitrary》 structures

《Let X be a type, then $\ldots$ 》

In set theory, we can quantify over all sets

We can also use Grothendieck universes: a set with strong closure property (e.g. $\cup_{i\in I} x_i$ is an element of U if I is in U and each x_i is in U) in order to form the set of all structures at a given universe

Universes

In type theory, a universe is a type the elements of which are types

Notion already present in the system AUTOMATH (N.G. de Bruijn)

Universes

The same limitation holds for the notion of elementary topos

This limitation holds also for the notion of *sheaves*

As soon as one wants to describe a ⟪sheaf⟫ of *structures* one needs to replace the notion of sheaf by the notion of *stack*

For instance if we let $F(V)$ be the collection of sheaves over V

We have a natural restriction map $F(V) \rightarrow F(W)$ for $W \subseteq V$

This does *not* define a sheaf but a stack: 3 by 3 condition for glueing and glueing is only unique *up to isomorphism*

Universes

If U is a universe and $T : U$ then T is a type

A function $B : A \to U$ defines a family of (U-)types over A

Using this, we can form non constant families of types

E.g. $F\ 0 = N, \qquad F\ (n+1) = F\ n \to N$

This defines a type family $F\ n$ for n of type N

We also have the *fundamental* example of the type family X for X of type U

And then we can form type family $X \times X$ or $X \times (X \to X)$ for X of type U

Universes

Natural to generalize the notion of function and product

-dependent function type $(x : A) \to B(x)$ also written $\Pi\ (x : A)\ B(x)$

-dependent sum $(x : A) \times B(x)$ also written $\Sigma\ (x : A)\ B(x)$

We recover $A \to B$ and $A \times B$ respectively if $B(x) = B$ is constant

Universes

We can then form a type of structures, e.g. $(X : U) \times X \times (X \to X)$

An element of this type is a triple (A, a, f) with

$A : U$

$a : A$

$f : A \to A$

Dependent types

These two operations

-$(x : A) \to B(x)$ $\qquad f \qquad \text{with} \qquad f\ x = b$

-$(x : A) \times B(x)$ $\qquad (a, b)$

are *derived* operations in set theory

The second one is relatively subtle (cf. Bourbaki, Bishop)

Here they are *primitive* operations on *family of types*

Dependent types

Logical operations can be reduced to construction on types following the dictionnary

$A \wedge B$	$A \times B = (x : A) \times B$
$A \rightarrow B$	$A \rightarrow B = (x : A) \rightarrow B$
$(\forall x : A) B(x)$	$(x : A) \rightarrow B(x)$
$(\exists x : A) B(x)$	$(x : A) \times B(x)$
$A \vee B$	$A + B$

Dependent types

de Bruijn (1967) notices that this approach is suitable for representation of mathematical proofs on a computer (AUTOMATH)

Proving a proposition is reduced to building an element of a given type

《This reminds me of the very interesting language AUTOMATH, invented by Dijkstra's colleague (and next-door neighbor) N. G. de Bruijn. AUTOMATH is not a programming language, it is a language for expressing proofs of mathematical theorems. The interesting thing is that AUTOMATH works entirely by type declarations, without any need for traditional logic! I urge you to spend a couple of days looking at AUTOMATH, since it is the epitome of the concept of type.》

D. Knuth (1973, letter to Hoare)

Dependent types

For simple type theory, or for set theory, we need to describe the logical rules and axioms

In the present system, this is part of the rules of term formations

E.g. we have $f : (A \to B \to C) \to B \to A \to C$ if we define $f\ u\ y\ x = u\ x\ y$

$u : A \to B \to C$

$y : B$

$x : A$

New laws for equality

Martin-Löf introduces (1973) a primitive notion of equality in dependent type theory

The ⟪proposition⟫ expressing the equality of a_0 and a_1 of type A is represented by a family of type $\mathsf{Id}\ A\ a_0\ a_1$

This equality is *typed*

Different equality for different A (compare with set theory)

This introduces a new way to form non constant families of types

New laws for equality

Since Id $A\ a_0\ a_1$ is itself a type, one can iterate this construnction

$$\mathsf{Id}\ (Id\ A\ a_0\ a_1)\ p\ q$$

This is the core of the connection with ∞-groupoid

New laws for equality

What are the rules of equality?

(1) Any element is equal to itself $1_a : \mathsf{Id}\ A\ a\ a$

(2) $C(a)$ implies $C(x)$ whenever we have $p : \mathsf{Id}\ A\ a\ x$

New laws for equality

What are the rules of equality?

(1) Any element is equal to itself $1_a : \mathsf{Id}\ A\ a\ a$

(2) $C(a)$ implies $C(x)$ whenever we have $p : \mathsf{Id}\ A\ a\ x$

So any $p : \mathsf{Id}\ A\ a\ x$ defines a ⟪transport⟫ function $t(p) : C(a) \to C(x)$

In particular we have $t(1_a) : C(a) \to C(a)$

It is natural to ask for the law $\mathsf{Id}\ C(a)\ (t(1_a)\ u)\ u$ for $u : C(a)$

This refines the law (2)

New laws for equality

The *new* law discovered by Martin-Löf (1973) can be expressed as the fact that in the type

$$(x : A) \times \mathsf{Id}\ A\ a\ x$$

which contains the element

$$(a, 1_a) : (x : A) \times \mathsf{Id}\ A\ a\ x$$

any element (x, p) is actually *equal* to this special element $(a, 1_a)$

New laws for equality

Usual formulation is

$$(x : A) \to (p : \mathsf{Id}\ A\ a\ x) \to C(a, 1_a) \to C(x, p)$$

which generalizes

$$(x : A) \to \mathsf{Id}\ A\ a\ x \to P(a) \to P(x)$$

New laws for equality

Let us «test» this law on the following example

Let S be the collection of «all» sets

We fix a set A and $T = (X : S) \times (A \to X)$

Two elements (B, g) and (C, h) of T are identified if we have an isomorphism $f : B \simeq C$ such that $h = f \circ g$

If $Q = (X : S) \times A \simeq X$ any element (X, f) of Q can be identified to (A, id) since $f = f \circ id$, and actually, this identification is uniquely determined

Q seen as a groupoid is equivalent to the groupoid with one object and one morphism

New laws for equality

Let us define

$\mathsf{isContr}\ T = \quad (t : T) \times ((x : T) \to \mathsf{Id}\ T\ t\ x)$

This describes when a collection is «equivalent» to a singleton

The new law of equality can be expressed as inhabitant of

$\mathsf{isContr}\ ((x : A) \times \mathsf{Id}\ A\ a\ x)$

for any type A and a element of A

Singleton types are contractible

$$\begin{tikzcd}
a \arrow[r, dash, "p"] \arrow[d, dash, "1_a"'] & x \arrow[d, dash, "p"] \\
a \arrow[r, dash, "1_a"'] & a
\end{tikzcd}$$

Any element x, p in the type $(x : A) \times \mathsf{Id}\ A\ a\ x$ is equal to $a, 1_a$

Loop space

《Indeed, to apply Leray's theory I needed to construct fibre spaces which did not exist if one used the standard definition. Namely, for every space X, I needed a fibre space E with base X and with trivial homotopy (for instance contractible). But how to get such a space? One night in 1950, on the train bringing me back from our summer vacation, I saw it in a flash: just take for E the space of paths on X (with fixed origin a), the projection $E \to X$ being the evaluation map: path $\to$ extremity of the path. The fibre is then the loop space of (X, a). I had no doubt: this was it! $\ldots$ *It is strange that such a simple construction had so many consequences.*》

J.-P. Serre, describing the 《loop space method》 introduced in his thesis (1951)

New laws for equality

It follows from these laws that any type has a ⟪groupoid⟫ structure

For instance, composition corresponds to transitivity of equality

The fact that equality is symmetric corresponds to the inverse operation

Hoffman-Streicher (1993) Lamarche (1991)

New laws for equality

These laws were discovered in 1973

Should equality be extensional?

Actually, how to express the extensionality axioms in this context?

An answer to this question is given by Voevodsky (2010)