Formalization of Mathematics and Dependent Type Theory

Thierry Coquand

Nordic Online Logic Seminar, 28/02/2022

This talk

-First part

Some formal systems used for representing mathematics on a computer

Set Theory/Type Theory

Hilbert-style/Natural Deduction

-Second part

Unexpected development for the formalism of dependent type theory: seems to be surprisingly well adapted for expressing concepts and proofs of *higher* topos theory

Formalisation of Mathematics

While trying to represent mathematics on a computer, one should understand something about the nature of mathematical objects

How to teach logic? First-order logic?

Not easy to explain free and bound variables and correct substitutions for the rule

$(\forall x \psi(x)) \ \rightarrow \ \psi(t/x)$

t has to be free for x in $\psi(x)$

One should then prove that this rule is semantically valid

Formalisation of Mathematics

A. Tarski *A simplified formalization of predicate logic with identity*, 1964

Two of the notion commonly used in describing the formalism of (first-order) predicate logic exhibit less simple intuitive context and require definitions more careful and involved than the remaining ones. These are the notion of a variable occuring free at a given place of a formula and the related notion of proper substitution... The relatively complicated character of these two notions is a source of certain inconveniences of both practical and theoretical nature; this is clearly experienced both in teaching an elementary cource of mathematical logic and in formalizing the syntax of predicate logic for some theoretical purposes.

Formalisation of Mathematics

This paper presents an analysis of the substitution $\psi(t/x)$

If we have equality, we can define it as $\forall x\ (x = t \rightarrow \psi)$

In this way, we avoid the problem of dealing with the notion of a term t free for a variable x in ψ

This analysis is used in the system Metamath, designed 1993 by the late N. Megill

Metamath, A Computer Language for Mathematical Proofs

Formalisation of Mathematics

How to present logical rules?

Hilbert-style: a proof is a finite sequence of formulae $\psi_1, \ \ldots \ , \psi_n$

Each formula is an instance of an axiom schema, or can be deduced by modus ponens from two previous formulae

Formalisation of Mathematics

With Hilbert-style, the formal proof has little connections with the way one reasons

E.g. one can prove $\varphi \to \varphi$ from the axiom schemas

$a \to b \to a$ and $(a \to b \to c) \to (a \to b) \to a \to c$

the proof has little to do with how we convince ourselves that $\varphi \to \varphi$ is valid

These are the 2 first axioms of Frege's system 1879

Formalisation of Mathematics

Natural Deduction was introduced by Gentzen (and independently by Jaśkowski) precisely in order to have a formal system closer to the intuitive notion of "convincing argument"

Wonderful analysis with discovery of a symmetry between introduction and elimination of logical connectives

Most logical laws are "naturally" expressed in this way

One surprise is that the *law of excluded middle* gets a *special* status in this representation

Formalisation of Mathematics

My starting point is this: The formalization of logical deduction, especially as it has been developped by Frege, Russell and Hilbert, is rather far removed from the forms of deduction used in practice in mathematical proofs. Considerable formal advantages are achieved in return.

In contrast, I intended first to set up a formal system which comes as close as possible to actual reasoning, The result was a "calculus of natural deduction" ("NJ" for intuitionist, "NK" for classical predicate logic). This calculus then turned out to have certain special properties; in particular, the "law of the excluded middle", which the intuitionists reject, occupies a special position."

Gentzen, *Investigation into logical deductions*, 1935, translation by M.E. Szabo

Formalisation of Mathematics

Not clear how to define formally the notion of proof tree introduced by Gentzen, and later used by Prawitz

In particular, the introduction rule for implication and forall is quite subtle

Hilbert-style is simpler to represent

Automath

Remarkable representation of natural deduction by N.G. de Bruijn in the system Automath

see archive papers https://www.win.tue.nl/automath/

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

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

Analysis of the way a mathematician carries "in his head" a reasoning, and discovers in this way a uniform treatment of *terms* and *proofs*

Automath

Representation of a statement

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* $f(x) > n$.

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

She/He can state $f(\pi) > 5$ by *applying* theorem 1 and by giving *in this order*

π, the proof (1), 5 the proof (2)

Automath

In Automath this will become

Corollary = theorem1(π, (1), 5, (2)) : $f(\pi) > 5$

Automath

A crucial notion in Automath, inspired from the notion of *block structure* in ALGOL 60, is the one of *context*

Sequence of variable declaration with their types and named hypotheses in an *arbitrary* order

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

Automath also had a primitive "sort" type

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

Automath

Automath used same notation $[x : A]M$ for typed abstraction $\lambda_{x:A}M$ and for dependent product $\Pi_{x:A}B$

One obtained then a quite minimal calculus

$$M, A \ ::= \ x \mid M \ M \mid [x : A]M \mid \text{type}$$

I will use

$$M, A \ ::= \ x \mid M \ M \mid \lambda_{x:A}M \mid \Pi_{x:A}B \mid \text{type}$$

Automath

One of the first example was equality on A : type

Eq : $A \to A \to$ type

refl : $\Pi_{x:A}$Eq x x

eucl : $\Pi_{x\ y\ z:A}$Eq x $z \to$ Eq y $z \to$ Eq x y

These were introduced as *primitives*

$\lambda_{x\ y:A}\lambda_{h:\mathsf{Eq}\ x\ y}$ eucl y x y (refl y) h

is then of type $\Pi_{x\ y:A}$ (Eq x y) $\to$ Eq y x

Automath

Uniform treatment of *terms* and *proofs*

Use of λ-calculus for representating proof trees

The term

$\lambda_{x\ y:A}\lambda_{h:\mathsf{Eq}\ x\ y}\mathsf{eucl}\ y\ x\ y\ (\mathsf{refl}\ y)\ h$

can be seen as a representation of what is going on in the head of a mathematician when she/he is following an argument

These proof terms are not "static" objects: one can instantiate them, reduce them and this also corresponds to recognizable mental operations

Deduction Theorem

The crucial rule corresponding to the *deduction* theorem is

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda_{x:A} M : \Pi_{x:A} B}$$

If x is not free in B then $\Pi_{x:A} B$ represents implication $A \to B$

This provides a representation of introduction rule for implication and forall quantification

Deduction Theorem

The converse is application

If N is of type $\Pi_{x:A} B$ then $N\ x$ is of type B

$$\frac{\Gamma \vdash N : \Pi_{x:A} B}{\Gamma, x : A \vdash N\ x : B}$$

Automath vs Metamath

Two very different approaches

Automath: deduction theorem built in

Metamath: Hilbert-style system

One can "simulate" the Deduction Theorem (Mario Carneiro) by proving systematically $\varphi \to \psi$, instead of proving simply ψ, where φ is a "varying" formula

Automath vs Metamath

One replaces the sequence $\psi_1, \ \ldots \ , \psi_n$ by $\varphi \to \psi_1, \ \ldots \ , \varphi \to \psi_n$

E.g. if ψ_k follows from ψ_i and $\psi_j = \psi_i \to \psi_k$ then one shows that $\varphi \to \psi_k$ follows from $\varphi \to \psi_i$ and $\varphi \to \psi_j = \varphi \to (\psi_i \to \psi_k)$

Since Frege had the axiom $(a \to b \to c) \to (a \to b) \to a \to c$ he probably understood this in 1879, and this understanding was somehow lost in Russell's and then Hilbert's presentation

Automath vs Metamath

Automath: based on λ-calculus with a constant/definition mechanism

Metamath: constants introduced with logical equivalence

Freek Wiedijk *Comparing mathematical provers*, 2003

Agda and Metamath at similar places: no automation

Automath vs Metamath

Metamath is *very* efficient

cf. Mario Carmeiro *Metamath, the cartesian theorem prover*, 2019

More work is needed for understanding how to represent efficiently the notion of definitions in Automath

Can we get an Automath implementation as efficient as the one of Metamath?

Simple type theory

In Metamath, most examples are within an axiom system for *set theory*

Another formalism used in proof assistants is the one of *Higher-Order Logic*

A variation of Alonzo Church 1940 *simple type theory*

The base types are bool (propositions) and ι (individuals)

We close them by function types

Quantification over *any* types; first-order logic is (and historically appeared as) the restriction where we quantify only over objects of type ι

Simple type theory

One gets in this way a "minimal" version of higher-order logic

Church added the axioms

-function extensionality

-unique choice $(\exists!_x \psi) \to \psi(\iota(\psi)/x)$

-strong choice $(\exists_x \psi) \to \psi(\epsilon(\psi)/x)$

He also suggested the possibility of having propositional extensionality: two logically equivalent propositions are equal

This was discussed in the introduction of the second edition of Principia Mathematica

Dependent type theory

Martin-Löf has suggested various *refinements* of simple type theory

This provides yet another motivation for Automath's uniform treatment of terms and proofs

Series of systems: 1971, 1972, 1975, 1979

One original motivation was the fact that simple type theory lacks the possibility of quantifying over an arbitary structure (like in set theory where we can quantify over any set)

Dependent type theory

A Theory of Types, February 1971, revised October 1971

On the strength of intuitionistic reasoning, talk at the Bucharest conference 1971, August 29 to September 4

First place where Martin-Löf hints at a connection of dependent type theory with computer science

Dependent type theory

Motivation of these works: J.-Y. Girard's proof of normalisation for second-order logic, and then higher-order logic

However finite types may not be enough

The simple theory of finite types, although proof theoretically quite strong, has some unnatural limitations (for example, it permits only finite iterations of the power operation) and, above all, it is not adequate for a formalization of mathematics that talk about arbitrary sets and not just sets of natural numbers, sets of sets of natural numbers, and so on.

Dependent type theory

Something very similar to this generalization of simple type theory in order for representing arbitrary sets and arbitrary structures appeared in the 60s when sheaf models appeared in algebraic geometry

Sheaves of functions/sheaves of structures (like collection of modules)

Problem of patching together modules instead of patching functions

Grothendieck topos/elementary topos/simple type system

Dependent type theory

1971: Martin-Löf suggested to add a type of all types $V \in V$

Also added primitive data types, type of natural numbers and type of ordinals

Independently, this introduction of primitive inductively defined data types was suggested in the paper of D. Scott on *Constructive Validity*, 1970

Dependent type theory

Also added $\Sigma_{x \in A} B(x)$ type of pairs a, b with $a \in A$ and $b \in B(a/x)$

This can be used to represent types of structures: e.g. $\Sigma_{X \in V} X$ represents the type of *pointed* types

Indeed, an element of this type is a pair X, a with $X \in V$ and $a \in X$

Dependent type theory

After $V \in V$ was shown inconsistent by Girard, the 1972 version replaced V by a *predicative* universe: like in category theory, distinction between small and large types

In 1975, he refined this by a cumulative sequence of universes

With natural numbers, the ordinal strength of the system is Γ_0 (conjectured by P. Hancock and later proved by P. Aczel and Feferman)

The last version 1979 introduces some strong form of function extensionality

Dependent type theory

The situation is subtle: these predicative systems are *weaker* than second-order arithmetic and simple type theory

On the other hand, the type system refines the one of HOL by introducing, besides function types, dependent function types and dependent sum types

All this is because of the introduction of the type V of small types

Dependent types

One can introduce an *impredicative* universe and gets in this way a strict extension of HOL, Th. C. 1985

This universe represents the type of propositions

This has been used to represent mathematics

-works reasonably well for *concrete* structures, like finite graphs or finite groups

-does not really work for abstract notions (e.g. arbitrary rings, non necessarily decidable)

Like for minimal HOL, one needs to add new principles

System Lean

The system adds

-classical logic

-propositional and functional extensionality

-strong form of choice

-quotient types (should be definable as type of equivalence classes?)

This system seems to work well in practice, cf. Liquid Tensor Experiment!

Mario Carneiro *The type system of Lean*, Master Thesis, CMU, 2019

Univalent Foundations

Other way to add propositional and function extensionality

Voevodsky noticed that the language of dependent type theory expresses really well notions of homotopy theory

$B(x),\ x : A$ fibration

$\Sigma_{x:A} B(x)$ total space of the fibration

$\Pi_{x:A} B(x)$ space of sections

Univalent Foundations

Voevodsky analysed the interpretation of type theory in term of spaces up to homotopy (Kan simplicial sets with a Quillen model structure)

He proved, with the help of Bousfield, that in this model, the universe should satisfy what is like a *dependent form of propositional extensionality*

This is the Axiom of Univalence

Weak form: two equivalent types are equal

This was the special case proved by Bousfield: given two Kan simplicial sets A and B, how to build a path between A and B given a weak equivalence $A \to B$

Univalent Foundations

He noticed (2011) that this axiom implies function extensionality!!

Also, in the groupoid model, the equality $(X, a) =_S (Y, b)$ for a type such as $S = \Sigma_{X:U} X$ is a bijection $e : X \simeq Y$ with an equality $e\ a =_Y b$, i.e. it is an *isomorphism* between the two *structures* X, a and Y, b

This suggested (Th. C., 2011) that univalence should imply that isomorphic structures are equal, and this was formally checked in Agda (with N.A. Danielsson) and later much refined by P. Aczel

Univalent Foundations

The fact that we can formulate such a refinement of *propositional* extensionality and that it implies *function* extensionality and identification of *isomorphic structures* is really remarkable, and should be considered as one of the biggest achievement in logic since the formulation of type theory by Russell 1908

Still, things are not completely understood yet: in order to represent quotients, Voevodsky had to use an operation of propositional truncation $\|A\|$ with changes of universe

The axiom of unique choice becomes then not only provable, but in a strong form which corresponds to mathematical practice: one can introduce a mathematical structure if it is uniquely characterised by a universal property

Sheaf models

In the 60s, rather surprising connections were discovered between sheaf models and intuitionistic logic

In a sheaf model over a topological space X, the truth values correspond to open subsets $U, V, \ldots$ and negation of U corresponds to the *interior* of the complement of U

$\neg\neg U$ is the the interior of the closure of U

Sheaf models (Grothendieck topos) are models of an *intuitionistic* version of higher-order logic

D. Scott stressed the fact that such models are natural generalisation of *forcing* models

Topos as generalised set theory

C. McLarty *The Uses and Abuses of the History of Topos Theory*, 1990

The notion of topos was introduced by Grothendieck

Lawvere-Tierney: notion of elementary topos 1970

Cartesian Closed Category with a subobject classifier

Model of *higher order logic* and not set theory

Dana Scott *A Proof of the Independence of the Continuum Hypothesis*, 1967

Topos as generalised set theory

His confidence in the project was strengthened by Dana Scott's work on Boolean valued models, which he heard about at a meeting that same spring at Oberwolfach. Even here it was not the set theoretic aspect of the work that caught Lawvere's attention but the logical aspect. He has said the independence proofs in ZF were less important to him than a paper in which Scott proved the continuum hypothesis independent of a kind of third order theory of the real numbers, because, Scott says: 'once one accepts the idea of Boolean values there is really no need to make the effort of constructing a model for full transfinite set theory' (Scott [1967], p. 109).

To Lawvere this seemed not only simpler than the version for ZF but more to the point.

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

Sheaf models and Universes

Thinking about Martin-Löf's motivation for extending simple type theory with a universe it is natural to look at the interpretation of universes in sheaf models

The same question holds for forcing models

Something quite interesting happens then

Generalization of forcing for universes?

How to interpret universes?

Type theory/set theory

Gödel/Tarski formulation of simple type theory: only types $0,\ 1,\ 2, \ldots$, with $n+1$ type of subsets of type n and 0 type of individuals

Set theory: start with 0 empty set and iterate power set transfinitely

See *A Formal Proof of the Independence of the Continuum Hypothesis*, J. M. Han, F. van Doorn, 2021

Sheaf models and Universes

Assuming that we have a universe $\mathcal{V}$ in the underlying set theory

How to interpret the first universe for the sheaf model over X?

We can define $F(U)$ as the collection of $\mathcal{V}$-small sheaves over U

This has a *presheaf* structure in a natural way

Is this a sheaf?

The answer is *no*: if we are given locally F_i sheaves over U_i that are compatible, there is not a *unique* way to patch them to a sheaf over U, only *unique up to isomorphisms*

Sheaf models and Universes

This question appeared in algebraic geometry in the 60s, and was a direct motivation of the notion of *stacks*

Cf. *Éléments de Géométrie Algébrique, 1*, 3.3.1, A. Grothendieck and J. Dieudonné

Similar questions appear when one wants to patch together *structures*, as opposed to elements (e.g. value of a function), e.g. in Weil's definition of manifold

Sheaf models and Universes

The natural notion is *isomorphism* and not strict *equality*

One introduces a notion of compatibility *up to isomorphisms*, the condition is then a *ternary* relation compatibility (instead of binary) and the patching is also unique *up to isomorphisms* and not up to equality

Sheaf models and Universes

This shows that the connection of dependent type theory with *groupoids* was somehow unavoidable

It was first hinted by F. Lamarche *A proposal about foundations I*, 1990

The use of this model for a proof relevant identity type was then discovered by M. Hofmann and Th. Streicher

Sheaf models and Universes

This problem appears when patching *structures*

It is natural to consider the problem of patching *groupoids* (collection of structures)

This will be a *quaternary* condition with uniqueness up to *equivalences*

Sheaf models and Universes

The notion of ∞-topos has been introduced to describe such generalisation of the notion of stacks

The language of type theory, which we have seen originates from similar motivation, appears in practice to be well-suited to express what is going on

Univalence corresponds to some principle known in the framework of ∞-topos the principle of *descent*

Cf. *On Voevodsky's univalence principle*, A. Joyal, Voevodsky Memorial Conference

Summary

First part: very different approaches for formalization of mathematics

Metamath: Hilbert-style

Automath: natural deduction

de Bruijn formulated an elegant way to represent natural deduction proof trees

Relevant and important question of *efficiency* of the Automath approach

Summary

Dependent type theory was also introduced as a refinement of simple type theory where one can represent arbitrary structures

Martin-Löf introduced for this the notion of *universe*, or type of small types

Voevodsky's univalent axiom is a natural refinement of propositional extensionality, with universe thought of as a type of propositions

Surprisingly, this principle implies function extensionality and the fact that isomorphic structures can be identified

The special status of the law of excluded-middle appears both in the setting of natural deduction and in sheaf models!

Summary

Dependent type theory provides a surpringly convenient language for *higher topos theory*, generalization of the notion of Grothendieck topos, where one can represent arbitrary structures, groupoids, and higher notions of groupoidsnnnb

Something corresponding to univalence was independently formulated in thisb setting

Both dependent type theory and higher topos theory were motivated by extending simple type system for representing arbitary structures

Supplement: Refinement of Natural Deduction

Via the proposition-as-types principle, $\Sigma_{x:A}B$ can be seen as a *strong* form of existential $\exists_{x:A}B$

This was noticed by Howard (1969): strong versus weak existential

Martin-Löf noticed that each logical notion gets a refined version of its elimination rule

Refinement of Natural Deduction

E.g. $A + B$, 1972

Instead of $(a \vee b) \to c$ follows from $a \to c$ and $b \to c$

$\mathsf{case}(u, v) : \Pi_{z:A+B} C(z)$ follows from $u : \Pi_{x:A} C(\mathsf{inl}\ x)$ and $v : \Pi_{y:B} C(\mathsf{inr}\ y)$

with natural computation rules (normalisation in natural deduction)

$\mathsf{case}(u, v)\ (\mathsf{inl}\ x) = u\ x$ $\qquad$ $\mathsf{case}(u, v)\ (\mathsf{inr}\ y) = v\ y$

This uses crucially the fact that one uses the formalism of *dependent types*

Refinement of Natural Deduction

Ex falso quodlibet $\bot \to A$ can be refined as $\Pi_{z:\bot} C(z)$, 1973

One also can refine the elimination for equality, 1973

$x =_A y \to R(x,y)$ if we have $\forall_{x:A} R(x,x)$

$\mathsf{J}(d) : \Pi_{x\ y:A} \Pi_{e:x=_A y} C(x,y,e)$ follows from $d : \Pi_{x:A} C(x,x,\mathsf{refl}\ x)$

$\mathsf{J}(d)\ x\ x\ (\mathsf{refl}\ x) = d\ x$