Philosophy of Mathematics

Joel David Hamkins and Ruizhi Yang

School of Philosophy

International Summer Session, Fudan University

Summer 2025

Setup of the Course

Peoples

- Main lecturer:

  Joel David Hamkins, John Cardinal O'Hara Professor of Logic at the University of Notre Dame

  - Research interests: set theory, particularly with forcing and large cardinals, philosophy of the mathematics, philosophical logic, etc.

Setup of the Course

Peoples

- Main lecturer:

  Joel David Hamkins, John Cardinal O'Hara Professor of Logic at the University of Notre Dame

  - Author of the book *Lectures on the Philosophy of Mathematics* (The MIT Press, 2021 and 上海人民出版社, 2025)

Setup of the Course

Peoples

- Main lecturer:

  Joel David Hamkins, John Cardinal O'Hara Professor of Logic at the University of Notre Dame

  - He is active on MathOverflow. He has earned the top-rated reputation score.
  - He will start delivering lectures from July 7.

Setup of the Course

Peoples

- Lecturer (for the first three meetings):

  Ruizhi Yang 杨睿之, Associate Professor at the School of Philosophy, Fudan University.

  Email: yangruizhi@fudan.edu.cn

- Teaching assistant:

  Boxiang Zeng 曾柏翔, 24210160034@m.fudan.edu.cn

Setup of the Course

Prerequisites

We do NOT intend to assume any particular background in philosophy or mathematics. However, some experience with mathematics will surely be helpful, and some familiarity with logic will be great.

Setup of the Course

Grading and Evaluation

- In-class discussion 40%: Students are encouraged to participate by asking questions and presenting their views.
- Final presentation 60%: The presentation should reflect the readings and the student's reflections.
  - Time: July 24 (and maybe July 22)
  - Format: TBA

Setup of the Course

Reference

- Joel David Hamkins, *Lectures on the Philosophy of Mathematics*, The MIT Press, 2021.
- Øystein Linnebo, *Philosophy of Mathematics*, Princeton University Press, 2017
- Stewart Shapiro, *Thinking about Mathematics: The Philosophy of Mathematics*, Oxford University Press, 2000.
- Jean van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press, 1967.
- Paul Benacerraf and Hilary Putnam (ed.), *Philosophy of Mathematics: Selected Readings*, Cambridge University Press, 1984.

Setup of the Course

Course website

- https://logic.fudan.edu.cn/event2025/jdh
- https://jdh.hamkins.org/

Setup of the Course

Scan to join the wechat group

Plan for the first three meetings

- General introduction to philosophy of mathematics
  - The philosophical challenges from mathematics
  - The search for a foundation of mathematics
- (If time permits) Topic that Joel might not cover
  - Type theory, proof assistants, and AI

The request for a practical foundation

- Verfification crisis
- Proof assistant
- A language mediating human beings and machines

  So that it can be used for reform the industry of mathematics

Verification crises

Kepler Conjecture

- Proof announced by Thomas Hales in 1998, with 250 pages of notes and 3 GB of computer programs, data and results.
- In 2003, *Annals of Mathematics* review: "99% certain"
- A formal proof was announced in 2014.
- In 2017, the formal proof (in HOL Light and Isabelle) was accepted by *Forum of Mathematics*

Verification crises

But, many were not as lucky as Thomas Hales.

Shinichi Mochizuki's Work on the ABC Conjecture

- A proof was announced by Shinichi Mochizuki in 2012, with "Inter-universal Teichmüller theory" in 750 pages
- In 2018, After long discussion with Mochizuki, Scholze and Stix wrote a report *Why abc is still a conjecture*.
- In 2021, Mochizuki's proof was published in RIMS, but have not gain broad acceptance.

Things can be otherwise

Proof assistant

Peter Scholze's Liquid Tensor Experiment

- In 2020, Peter Scholze raised a challenge to formally verify a central and complex theorem in his and Dustin Clausen's theory of condensed mathematics. Due to the proof's immense complexity, Peter Scholze was not completely certain of its correctness

Proof assistant

I spent much of 2019 obsessed with the proof of this theorem, almost getting crazy over it. In the end, we were able to get an argument pinned down on paper, but I think nobody else has dared to look at the details of this, and so I still have some small lingering doubts.

(Scholze, *Liquid tensor experiment*)

Proof assistant

with this theorem, the hope that the condensed formalism can be fruitfully applied to real functional analysis stands or falls. I think the theorem is of utmost foundational importance, so being 99.9% sure is not enough.

(Scholze, *Liquid tensor experiment*)

Proof assistant

Peter Scholze's Liquid Tensor Experiment

- In May 2021, Half a year later, the formalized proof of Theorem 9.4 was announced, which concludes the first part of the project.
- A year and a half later, in 2022, it was announced completed by the Lean community. A human-readable blueprint was included.

Proof assistant

Theorem 9.4 is an extremely technical statement, whose proof is however the heart of the challenge, and is the only result I was worried about. So with its formal verification, I have no remaining doubts about the correctness of the main proof.

(Scholze, *Half a year*)

Proof assistant

The Lean Proof Assistant was really that: An assistant in navigating through the thick jungle that this proof is. Really, one key problem I had when I was trying to find this proof was that I was essentially unable to keep all the objects in my "RAM", and I think the same problem occurs when trying to read the proof.

(Scholze, *Half a year*)

Proof assistant

Lean always gives you a clear formulation of the current goal, and Johan confirmed to me that when he formalized the proof of Theorem 9.4, he could —with the help of Lean —really only see one or two steps ahead, formalize those, and then proceed to the next step. So I think here we have witnessed an experiment where the proof assistant has actually assisted in understanding the proof.

(Scholze, *Half a year*)

Proof assistant

Fermat's Last Theorem (FLT)

- The first broadly accepted proof of it is announced by Sir Andrew Wiles in 1993.
- The original proof uses Grothendieck's universe, which equivalents to the existence of an inaccessible cardinal
- It was built upon a gigantic edifice of 20th century mathematics.

Proof assistant

There are programs aiming at

- Proof FLT in a weak arithmatical theory (Reverse mathematics)
- Reduce FLT to claims which were known to mathematicians by the end of the 1980s (The FLT Project in Lean community)

Mediating human beings and machines

Formal proof assistants can be used to verify proofs (as well as the output of large language models), allow truly large-scale mathematical collaborations, and help build data sets to train te aforementioned macine learning algorithms.

(Tao, AMS Colloquium Lectures)

Mediating human beings and machines

One notable feature of proof formalization projects is that they lend themselves to large collaborations that do not require high pre-established levels of trust.

(Tao, AMS Colloquium Lectures)

Type theory

Type Theory is behind

- RCoq
- Lean
- Agda

Type theory

A brief introduction to modern type theory based on

The HoTT Book

Type theory

Basic judgements of type theory

$$a : A \qquad \text{(well-typed)}$$

The key ideas

- propositions as types
- proof by construction
- construction according to the RULEs

Type theory

- Definitional equality (Another form of basic judgements)

  Example: $f(x) \equiv x + x$ (in metalanguage)

- Propositional equality

  Example: $p : a =_A b$

Type theory

The universes of types

Like set theory, in type theory, a type is also an element of a type. We let $\mathcal{U}$ to be the type (universe) of all types. To avoid Russell's paradox. We can, for example, assume that we have a cumulative hierarcy $\mathcal{U}_0 : \mathcal{U}_1$, $\mathcal{U}_1 : \mathcal{U}_2$, ... Usually, we write $a : \mathcal{U}$ (or say a is a type) to mean $a : \mathcal{U}_i$ for some i.

Type theory

the unit type and the empty type

- The unit type $\mathbf{1}$ has a particular element $\star$, so $\star : \mathbf{1}$
- We also have an empty type $\mathbf{0}$ which is not inhabited. So $a : \mathbf{0}$ is always ill-typed

Type theory

the coproduct type

- If we have a type A and a type B, then we have the coproduct type $A + B$ (type formation rule)

  The set-theoretical intuition is that $A + B$ is the disjoint union of set A and B.

- Given $a : A$, we have $\mathsf{inl}(a) : A + B$, and given $b : B$, we have $\mathsf{inr}(b) : A + B$ (element constructor)

The type $\mathbf{2}$

- We define $\mathbf{2} \equiv \mathbf{1} + \mathbf{1}$
- We also define $0_{\mathbf{2}} \equiv \mathsf{inl}(\star)$, and $1_{\mathbf{2}} \equiv \mathsf{inr}(\star)$
- You can tell that by definition, $0_{\mathbf{2}} : \mathbf{2}$, and $1_{\mathbf{2}} : \mathbf{2}$

Type theory

- You can try to construct type $\mathbf{3}$, $\mathbf{4}$, ...
- But these cannot give us something like a type $\mathbb{N}$

Function types

- Given a type A and a type B, we have a type

$$A \to B$$

- Intuitively, elements in $A \to B$ are functions from A to B.

  For example:

$$\mathsf{id}_N \equiv (\lambda x : \mathbb{N}).x : \mathbb{N} \to \mathbb{N}$$

Type theory

λ-abstraction

λ-abstractions are of the form $(\lambda x : A)\ t$, where t is a term possibly have x occurring free in it. They are in the metalanguage, telling us how to construct an element in a (dependent) function type. For example,

$$(\lambda x : \mathcal{U})\ x + \mathbf{1} : \mathcal{U} \to \mathcal{U}$$

Type theory

Dependent function type

Given type A and type $B : A \to \mathcal{U}$, we have type

$$\prod_{x:A} B(x)$$

In set-theoretical intuition, this is the set of all functions f, such that the domain of f is A, and $f(x) \in B(x)$ for any $x \in A$

Type theory

Dependent pair type

Given type A and $B : A \to \mathcal{U}$, we have type

$$\sum_{x:A} B(x)$$

If $a : A$ and $b : B(a)$, then $(a, b) : \sum_{x:A} B(x)$

Type theory

As the function types are special dependent function types, we also hhave

Product type

Given type A and B (or $B : A \to \mathcal{U}$ is a constant function), we have type

$$A \times B$$

And if $a : A$ and $b : B$, then $(a, b) : A \times B$

BHK interpretation

Brouwer–Heyting–Kolmogorov interpretation

- A proof of $P \wedge Q$ is a pair (a, b) where a is a proof of P and b is a proof of Q.
- A proof of $P \vee Q$ is either $(0, a)$ where a is a proof of P or $(1, b)$ where b is a proof of Q.
- A proof of $P \to Q$ is a construction that converts a (hypothetical) proof of P into a proof of Q.

BHK interpretation

Brouwer–Heyting–Kolmogorov interpretation

- A proof of $(\exists x \in S)(Px)$ is a pair (x, a) where x is an element of S and a is a proof of Px.
- A proof of $(\forall x \in S)(Px)$ is a construction that converts an element x of S into a proof of Px.
- The formula $\neg P$ is defined as $P \to \bot$, so a proof of it is a construction that converts a proof of P into a proof of $\bot$.
- There is no proof of $\bot$ (absurdity).

Type theory

Propositions as types

- True $\equiv \mathbf{1}$
- False $\equiv \mathbf{0}$
- $A \wedge B \equiv A \times B$
- $A \vee B \equiv A + B$
- $A \to B \equiv A \to B$

Type theory

Propositions as types

- $\neg A \equiv A \to \mathbf{0}$
- $(\forall x \in A)Px \equiv \prod_{x:A} P(x)$
- $(\exists x \in A)Px \equiv \sum_{x:A} P(x)$

Type theory

Proof by construction (of elements in types) in Lean 4

A very first impression

AI and Mathematics

Philosophical Reflections

- Can mathematicians be replaced by machines?
  - Is mathematics ultimately empirical?
  - Can machines perform empirical reasoning better than humans?
  - Can machines learn to recognize what constitutes good mathematical work?
- To what extent can machines assist us in doing mathematics? Is there any fundamental obstacle to the full formalization of mathematics?

Next

Joel David Hamkins