Philosophy of Mathematics as a Design Science

Jeremy Avigad

Department of Philosophy and
Department of Mathematical Sciences
Carnegie Mellon University

March 2018

Epistemological questions

Since Plato, the philosophy of mathematics has been concerned with:

- the nature of mathematical objects, and
- the appropriate justification for mathematical knowledge.

But we employ other normative judgments as well:

- some theorems are interesting
- some questions are natural
- some concepts are fruitful, or powerful
- some proofs provide better explanations than others
- some historical developments are important
- some observations are insightful

. . . and so on.

The problem of multiple proofs

On the standard account, the value of a mathematical proof is that it warrants the truth of the resulting theorem.

Why, then, do we often value a new proof of a previous established theorem?

For example, Gauss published six proofs of the law of quadratic reciprocity in his lifetime, and left us two unpublished versions as well.

Franz Lemmermeyer has documented 233 proofs (available online, with references).

The problem of multiple proofs

This question not new. For example:

> *It might be said: "—that every proof, even of a proposition which has already been proved, is a contribution to mathematics". But why is it a contribution if its only point was to prove the proposition? Well, one can say: "the new proof shews (or makes) a new connexion". — Wittgenstein, Remarks on the Foundations of Mathematics, III–60*

Indeed, it is *not* a great mystery. There is a lot we can say about what we learn from different proofs.

But the philosophy of mathematics has had relatively little to say about the matter.

The problem of conceptual possibility

It is often said that some mathematical advance was "made possible" by a prior conceptual development.

For example, Riemann's introduction of the complex zeta function and the use of complex analysis made it possible for Hadamard and de la Vallée Poussin to prove the prime number theorem in 1896.

What is the sense of "possibility" here?

Intuition: a certain *understanding* guides us.

Epistemological questions

What the questions have in commmon:

- They have a generally epistemological flavor, involving "knowledge" or "understanding."
- They invoke normative assessments.

This is a starting point for philosophical inquiry.

Outline

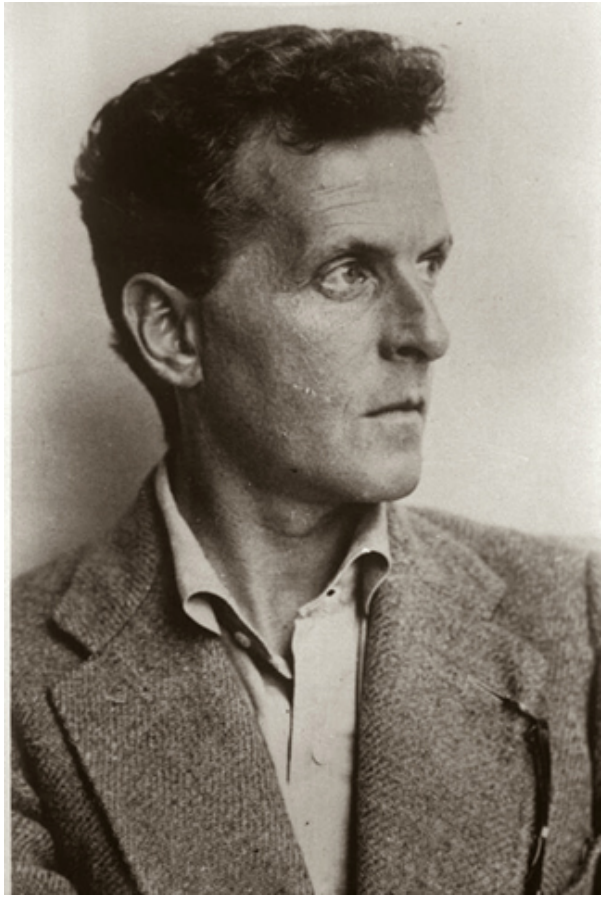
Overview:

- General epistemological questions
- Mathematics from a design perspective
- Towards a theory of mathematical understanding
- Strategies
  - Look to mathematical practice
  - Look to interactive theorem proving
  - Look to the history of mathematics
- Modularity in mathematics

I learned empirically that this came out this time, that it usually does come out; but does the proposition of mathematics say that? . . . The mathematical proposition has the dignity of a rule.

So much is true when it's said that mathematics is logic: its moves are from rules of our language to other rules of our language. And this gives it its peculiar solidity, its unassailable position, set apart.

— Ludwig Wittgenstein

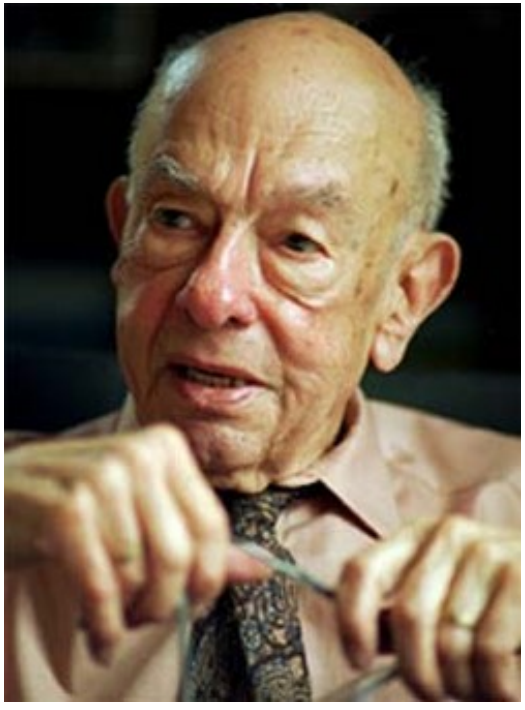


. . . it seemed to me one of the most important tasks of philosophers to investigate the various possible language forms and discover their characteristic properties. While working on problems of this kind, I gradually realized that such an investigation, if it is to go beyond common-sense generalities and to aim at more exact results, must be applied to artificially constructed symbolic languages. . . . Only after a thorough investigation of the various language forms has been carried through, can a well-founded choice of one of these languages be made, be it as the total language of science or as a partial language for specific purposes.

— Rudolf Carnap

Physical objects, small and large, are not the only posits. . . . the abstract entities which are the substance of mathematics. . . are another posit in the same spirit. Epistemologically these are myths on the same footing with physical objects and gods, neither better nor worse except for differences in the degree to which they expedite our dealings with sense experiences.

— W. V. O. Quine



“When I use a word,” Humpty Dumpty said in rather a scornful tone, “it means just what I choose it to mean — neither more nor less.”

“The question is,” said Alice, “whether you can make words mean so many different things.”

“The question is,” said Humpty Dumpty, “which is to be master — that’s all.”

— Lewis Carroll

Philosophical puzzles

- Mathematics tells us about the world, but not vice-versa.
- Mathematical objects are not located in space or time.
- Mathematics delivers (near?) certainty.

Lowbrow answers

- Mathematics is governed by mathematical norms.
- We learn these norms from parents, teachers, . . .
- We come to have mathematical knowledge by following these norms correctly.

But why are the norms the way they are, and why do they tell us anything about the world?

The linguistic turn

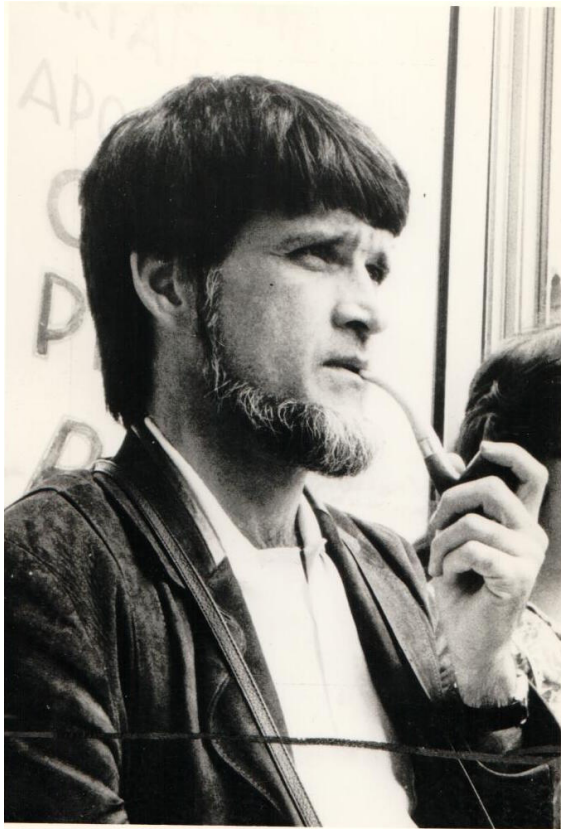
Mathematics is part of our language.

- Linguistic norms govern the way we describe the world.
- We have adopted these norms because they are useful.

These themes (with variations) occur throughout Wittgenstein, Carnap and the Logical Positivists, and Quine.

Only empirical explanation is possible for why we have come to accept the basic principles that we do and why we apply them as we do—for why we have mathematics and why it is at it is. But it is only within the framework of mathematics as determined by this practice that we can speak of mathematical necessity. In this sense, which I believe Wittgenstein was first to fully grasp, mathematical necessity rides on the back of empirical contingency.

— William Tait



Outline

Overview:

- General epistemological questions
- Mathematics from a design perspective
- Towards a theory of mathematical understanding
- Strategies
  - Look to mathematical practice
  - Look to interactive theorem proving
  - Look to the history of mathematics
- Modularity in mathematics

Towards a theory of mathematical understanding

General picture:

- Beyond knowledge, we look to mathematics for modes of *understanding*.
- Understanding involves not just factual knowledge, but something more dynamic: ways of proceeding, modes of analysis, capacities for thought.
- We value mathematical resources for conferring understanding.
- Some mathematical resources are overtly syntactic: definitions, theorems, proofs, questions.
- These give rise to resources that are harder to characterize precisely: concepts, methods, heuristics, intuitions, ...

A methodological stance

To make progress, we have to pick a methodological framework:

- a way of thinking about mathematics
- a language for talking about the objects of mathematical understanding
- a way of posing questions precisely (or at least trying to)
- precise, disciplined ways of answering them

We just have to do it, and see what happens.

A methodological stance

We want a philosophical theory of mathematical understanding that

- is coherent
- is satisfying
- can inform (and is informed by) other pursuits:
  - history of mathematics
  - interactive theorem proving and automated reasoning
  - psychology and cognitive science
  - mathematics education
  - mathematics itself

I will make some recommendations here.

Recommendations

First recommendation: stay grounded in syntax.

What characterizes mathematics with respect to other scholarly disciplines is its level of rigor: there are precise norms that dictate how to make meaningful mathematical claims, and how to establish their truth.

We can (and have) studied these norms in syntactic terms, with great success.

Definitions, theorems, proofs, conjectures, questions, and the like — the "literature" — constitute the starting data.

The more nebulous objects of understanding — concepts, methods, intuitions, etc. — are manifested in the linguistic artifacts.

Recommendations

Second recommendation: think of the philosophy of mathematics as a design science, like automative engineering.

A closer look at the syntactic components of mathematics — definitions, theorems, proofs, theories, and so on — shows them to be highly structured objects.

When one studies the history of mathematics, or tries to model *real* mathematical proofs formally, one has the sense that mathematical language is beautifully *designed* to extend our cognitive reach, make it possible for us to solve increasingly more difficult problems, construct more elaborate proofs.

What are the general principles?

Recommendations

Third recommendation: start with more specific, focused projects.

I will discuss three strategies for making progress:

- look to the everyday practice of mathematics
- look to the history of mathematics
- look to interactive theorem proving

Strategies

First strategy: look at ordinary mathematical proofs.

- What are the (inferential and communicative) norms that are in play?
- What cognitive capacities that are presupposed by their comprehensibility?

Compare alternative proofs, or textbook presentations, of the same theorem. Explain

- the structuring of information, and
- the understanding or expertise that is conveyed.

We need to rely on what mathematicians *do* rather than their self assessments.

Strategies

Second strategy: look to the history of mathematics.

Find an important historical development (what Ken Manders calls a "big deal difference").

This suggests that we were in

- a certain epistemological state beforehand, and
- a certain epistemological state after,

and that they are different in some important way.

Explain the difference.

Strategies

Third strategy: look to interactive theorem proving and automated reasoning.

Formal verification involves the use of formal methods to verify correctness, for example:

- verifying that a circuit description, an algorithm, or a network or security protocol meets its specification; or
- verifying that a proof of a mathematical theorem is correct.

"Interactive theorem proving" is one important approach.

Strategies

Working with a proof assistant involves conveying enough information to the system to confirm that there is a formal axiomatic proof.

In fact, most proof systems actually construct a formal proof object, a complex piece of data that can be verified independently.

"Proof languages" provide expressive models of ordinary mathematical language, designed to convey knowledge (and expertise) efficiently.

Understanding what is needed to develop mathematics formally provides insight into how the informal languages work as well.

Interactive theorem proving

```
theorem PrimeNumberTheorem:
  "(%n. pi n * ln (real n) / (real n)) ----> 1"

!C. simple_closed_curve top2 C ==>
  (?A B. top2 A /\ top2 B /\
    connected top2 A /\ connected top2 B /\
  ~(A = EMPTY) /\ ~(B = EMPTY) /\
   (A INTER B = EMPTY) /\ (A INTER C = EMPTY) /\
       (B INTER C = EMPTY) /\
      (A UNION B UNION C = euclid 2)

!d k. 1 <= d /\ coprime(k,d)
  ==> INFINITE { p | prime p /\ (p == k) (mod d) }
```

Interactive theorem proving

```
Theorem Sylow's_theorem :
  [/\ forall P,
      [max P | p.-subgroup(G) P] = p.-Sylow(G) P,
      [transitive G, on 'Syl_p(G) | 'JG],
      forall P, p.-Sylow(G) P ->
        #|'Syl_p(G)| = #|G : 'N_G(P)|
   &  prime p -> #|'Syl_p(G)| %% p = 1%N].

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

Theorem simple_odd_group_prime (gT : finGroupType)
    (G : {group gT}) :
  odd #|G| → simple G → prime #|G|.
```

Interactive theorem proving

```
theorem (in prob_space) central_limit_theorem:
  fixes X :: "nat ⇒ 'a ⇒ real"
    and μ :: "real measure"
    and σ c :: real
    and S :: "nat ⇒ 'a ⇒ real"
  assumes X_indep: "indep_vars (λi. borel) X UNIV"
    and X_integrable: "⋀n. integrable M (X n)"
    and X_mean: "⋀n. expectation (X n) = c"
    and σ_pos: "σ > 0"
    and X_square_integrable:
        "⋀n. integrable M (λx. (X n x)²)"
    and X_variance: "⋀n. variance (X n) = σ²"
    and X_distrib: "⋀n. distr M borel (X n) = μ"
  defines "S n x ≡ ∑ i<n. X i x"
  shows "weak_conv_m (λn. distr M borel
       (λx. (S n x - n * c) / sqrt (n*σ²)))
    std_normal_distribution"
```

Interactive theorem proving

Challenges:

- Modeling mathematical assertions in a natural way.
- Modeling mathematical proof in a natural way.
- Modeling mathematical expertise, and filling in "straightforward" inferences automatically.
- Managing large libraries of information.
- Verifying long computations.

Lessons

Some of the things we have learned:

- Language is important.
- Notation is important.
- Definitions are important.
- Organization is important.
- Structure is important.
- Infrastructure is important.
- Matching and unification are important.
- Indexing and retrieval are important.
- Methods of reasoning are important.
- Heuristics are important.

The philosophy of mathematics should help us better understand how, and why.

Lessons

Designing a theorem prover involves designing a language (in a broad sense):

- axioms, rules
- syntax, notation
- semantics
- idioms
- concepts
- theories

A theorem prover and its libraries can be well designed, or poorly designed.

The same is true of a piece mathematics.

Outline

Overview:

- General epistemological questions
- Mathematics from a design perspective
- Towards a theory of mathematical understanding
- Strategies
  - Look to mathematical practice
  - Look to interactive theorem proving
  - Look to the history of mathematics
- Modularity in mathematics

Modularity in mathematics

Many important philosophical gains are focused: understanding a historical development, or recognizing an important inferential pattern.

In the time remaining, however, I will discuss one general theme: the value of modularity.

Modularity in mathematics

The term "modular" is a term of art in biology, computer science, business administration, architecture, neuroscience, cognitive science, philosophy of mind, ...

Thesis: Mathematical knowledge tends to be structured in modular ways.

(And we can be precise about how, and why.)

Modular systems

Herbert Simon, "The Architecture of Complexity," 1962, spoke of "nearly decomposible" systems rather than modular ones.

Modularity has been studied with respect to:

- biology
- social organizations (like a business)
- hardware design
- software design
- architecture
- the mind

Modular systems

Roughly, a complex system is said to be *modular* to the extent it has the following features:

- The system is divided into *components*, or *modules*, with *dependencies* between them.
- The division supports *abstraction*: the function of the components can be described with respect to the behavior of the entire system, without reference to the particular *implementation*.
- Dependencies between modules are kept small, and mediated by precise *specifications*, or *interfaces*.
- Dependencies within a module may be complex, but, due to *encapsulation* or *information hiding*, these are not visible outside the module.

Modular systems

A modular design is often claimed to bring certain benefits:

- *Comprehensibility:* makes it easier to understand, explain, and predict.
- *Independence:* allows the components of a system to be built and tested independently.
- *Reliability and robustness:* makes it easier to find and correct errors.
- *Flexibility:* makes it easier to change and adapt.
- *Reuse:* components that prove successful in one system can be used in others.

These are features we want our mathematics to have.

Modularity in computer science

Since the 1970's, modularity has been a central goal in software design:

- Large programs should be divided into independent modules.
- A *module* is a body of code with a well-defined *interface*. The interface specifies what procedures the user can call from the outside, what data these procedures expect, what data these procedures return, what state information the module keeps track of, and how procedural calls change the state.
- The internal workings of the code can otherwise largely be ignored; in particular, code that interacts through the interface is guaranteed to work even if the implementation changes.

From programs to proofs

The dialectic:

- The language of a proof assistant models informal mathematics.
- Text in such a language is a form of code.
- We know (more or less) how to talk about modularity in code.
- So it makes sense to talk about modularity in formal libraries.
- Insofar as these model informal mathematics, we can speak of modularity in mathematics.

Modularity in the wild

In everyday mathematics, modularity is everywhere:

- Books are divided into chapters.
- Proofs are broken down to lemmas.
- Subjects and bodies of knowledge are broken down into smaller disciplines.

Concrete examples can help us think about how the notions play out.

Congruence

Definition. If x and y are integers, say x *divides* y, written $x \mid y$, if there is an integer z such that $y = xz$.

Definition. If m is another integer, say x *is congruent to* y *modulo* m, written $x \equiv y \pmod{m}$, if $m \mid x - y$.

Let us consider a toy, but illustrative, example:

Proposition. If $x \equiv y \pmod{m}$, then $x^3 + 3x + 7 \equiv y^3 + 3y + 7 \pmod{m}$.

Congruence

Proof. Unpacking definitions, we have $x \equiv y \pmod{m}$ if and only if $x = y + mz$ for some z. Then

$$\begin{aligned} x^3 + 3x + 7 &= (y + mz)^3 + 3(y + mz) + 7 \\ &= y^3 + 3y^2mz + 3ym^2z^2 + m^3z^3 + 3y + 3mz + 7 \\ &= y^3 + 3y + 7 + m(3y^2z + 3ymz^2 + m^2z^3 + 3z) \end{aligned}$$

which shows that $x^3 + 3x + 7 \equiv y^3 + 3y + 7 \pmod{m}$. □

Of course, this doesn't scale.

More significantly, it breaks abstraction.

Congruence

Proposition. Let x, y, and z be integers.

1. $x \mid x$.
2. If $x \mid y$ and $y \mid z$ then $x \mid z$
3. If $x \mid y$ and $x \mid z$, then $x \mid y + z$.
4. If $x \mid y$, then $x \mid zy$.
5. $x \mid 0$.

Proof. For 1, we have $x = x \cdot 1$. For 2, if $y = xu$ and $z = yv$, then $z = x(uv)$. For 3, if $y = xu$ and $z = xv$, then $y + z = x(u + v)$. For 4, if $y = xu$, then $zy = x(zu)$. For 5, take $y = x$ and $z = 0$ in 3. □

This is the only place where we need to unfold the definition of $|$.

Congruence

Proposition.

1. $\equiv$ is an equivalence relation.
2. If $x_1 \equiv y_1 \pmod{m}$ and $x_2 \equiv y_2 \pmod{m}$ then $x_1 + x_2 \equiv y_1 + y_2 \pmod{m}$.
3. If $x_1 \equiv y_1 \pmod{m}$ and $x_2 \equiv y_2 \pmod{m}$ then $x_1 x_2 \equiv y_1 y_2 \pmod{m}$.
4. If $x \equiv y \pmod{m}$, then $x^n \equiv y^n \pmod{m}$ for every natural number n.

It follows that if $p(x)$ is any polynomial with integer coefficients and $x \equiv y \pmod{m}$, then $p(x) \equiv p(y) \pmod{m}$.

Congruence

In the refactored version:

- the existential quantifier in "divides" encapsulates data.
- the proofs about congruence respect that interface.

Benefits of the refactoring:

- The proof is easier to understand.
- The properties of divisibility and congruence are reusable.
- The result is more general.

Think about what is encapsulated with $\lim_{x \to a} f(x) = b$.

Algebraic abstraction and other strategies support modularity on the larger scale.

Modularity in mathematics summarized

It is generally understood that modularity brings benefits to software design:

- understandability
- reliability and robustness
- independence
- flexibility and adaptability
- generalizability and reuse

The notions carry over to mathematics.

For more detail, see "Modularity in mathematics," to appear in the *Review of Symbolic Logic*.

Outline

Overview:

- General epistemological questions
- Mathematics from a design perspective
- Towards a theory of mathematical understanding
- Strategies
  - Look to mathematical practice
  - Look to interactive theorem proving
  - Look to the history of mathematics
- Modularity in mathematics

Concluding remarks

We care about mathematics.

- We subject our children to countless hours of mathematical training.
- We put a lot of faith in mathematical results.
- We applaud mathematical achievements.

The subject deserves philosophical study that helps us understand what it means to do mathematics, and helps us do it better.