The Design of Mathematical Language

Jeremy Avigad

September 2021

Carnegie Mellon University

Thinking about mathematical language

Learning how to speak about mathematical objects properly is an important part of learning to do mathematics.

The outward manifestations of mathematical thought are found in the way we communicate mathematics to others.

Let's think about mathematical language and how it works.

Thinking about mathematical language

Formal languages and formal systems are helpful.

There are variations on:

- set theory
- simple type theory
- dependent type theory
- categorical foundations

There is a sense in which they specify too little, and there is a sense in which they specify too much.

Thinking about mathematical language

The fact that any conventional mathematical text can be expressed in any of those systems shows that the level of axiomatic primitives and rules is too fine-grained.

Formal axiomatic systems should be viewed *implementations* of something more abstract.

Thinking about mathematical language

There is also a sense in which formal systems specify too little.

Proof assistants have to:

- disambiguate notation
- manage libraries of theorems and definitions
- keep track of algebraic structures and relationships between them
- provide convenient manners of expression
- support efficient inference

Conventional formal logic doesn't say anything about how mathematics does all that effectively.

Thinking about mathematical language

Formal languages provide a low-level description.

Other approaches (e.g. Ganesalingam) approach mathematics from the perspective of natural language.

Here we won't be concerned with grammatical and stylistic variation.

Question: what are the features of mathematical language that are essential to supporting mathematical activity?

In other words, we'll be interested in the specifically mathematical features of mathematical language.

Thinking about mathematical language

Mathematical language is:

- more expressive than conventional formal systems
- more regimented than natural language

Goal: provide a *semiformal* description.

- Clarify the design specifications that proof assistants ought to meet.
- Idealize away the specifics of the implementation.

Philosophical orientation

Logic and philosophy of mathematics have focused on *justification*, clarifying what is *allowed*.

It is also important to understand what is *desired*:

- what makes for a good definition
- what makes for a natural question
- what makes for a promising inferential strategy

Mathematics requires us to carry out complex tasks, creatively, efficiently, and reliably.

We should try to understand how mathematical language helps us do that.

Philosophical orientation

Another central thesis: it is helpful to view mathematical language as a *designed* language.

The language has evolving to serve a purpose.

It provides a means for:

- managing information
- making data salient when it is needed
- suppressing it when it is a distraction

What are the design principles at play?

Methodological orientation

Let's try to be scientific about it.

How to we assess the success or accuracy of our description?

There are two sources of data.

Methodological orientation

Let's try to be scientific about it.

How to we assess the success or accuracy of our description?

There are two sources of data.

The first is the mathematical literature.

That is *a lot* of data: textbooks, journal articles, etc.

Methodological orientation

The second source of data: contemporary proof assistants.

Every proof assistant provides languages for defining objects, stating theorems, and proving them.

These are designed to model ordinary mathematics.

The experience of designing and working with them can help us make sense of the mathematical literature.

Reflection on the mathematical literature can help us design better proof assistants.

Table of contents

Table of contents

Sorts

A mathematical object is always an object of some *sort*.

> *Proposition 1. On a given straight line to construct an equilateral triangle.*
> *Let AB be the given finite straight line.*
> *Thus it is required to construct an equilateral triangle on the straight line AB.*
> *With centre A and distance AB let the circle BCD be described;*
> *again, with centre B and distance BA let the circle ACE be described;*
> *and from the point C, in which the circles cut one another, to the points A, B let the straight lines CA, CB be joined.*

Sorts

Objects are introduced as objects of some sort.

"Let X be locally convex topological vector space and F a closed convex subset."

"Let p and q be odd primes."

"Let k be a field, and let $k[x] = k[x_1, \ldots, x_n]$ be a finitely generated ring over k. Let $\varphi : k \to L$ be an embedding of k into an algebraically closed field L."

"Let $\mathfrak{M}$ be a maximal ideal of $k[x]$. Let σ be the canonical homomorphism $\sigma : k[x] \to k[x]/\mathfrak{M}$."

"Let $\tau_2 = n$; let t_1 be the smallest k such that $X_k \geq \alpha$, if there is one, and n otherwise."

Sorts

Variables generally range over some fundamental sort.

But it's not always clear what is most fundamental:

- equilateral triangle vs. triangle.
- prime number vs. number.
- finite straight line vs. straight line.
- nonnegative integer vs. integer vs. number.
- maximal ideal of R vs. subset of R vs. set.

In axiomatic set theory, there is only one fundamental sort.

Sorts

This is not unique to mathematics.

Aristotle distinguished between essential vs. accidental properties:

- Socrates is human.
- Socrates is short.

But the distinction has particular effects in mathematics.

Sorts

Specifically mathematical uses:

- To disambiguate notation (x^y, $x \cdot y$)
- To infer meaning (being maximal, surjective, continuous, a homomorphism, essentially bounded)
- To justify an expression as meaningful ($\sqrt{x}$, $1/x$)
- To support immediate inference (composition of functions vs. composition of morphisms)
- To support higher-level inference and heuristics.

Sorts

Note: this doesn't speak directly to the superiority of any one foundation over another.

It does speak in favor of:

- gaining clarity on how sortedness is introduced, tracked, and used in ordinary mathematical texts, and
- implementing languages that support it.

Numbers

We think of numbers like $0, 1, 2, \ldots$ as being concrete.

In Euclid, number is a multiplicity of units. The smallest one is 2.

In Euclid, and throughout early algebra, mathematicians only compared homogenous magnitudes.

In modern mathematics, we can multiply the number of planets by the length of the Suez canal, and add the number of children I have.

Let's not forget that this is a substantial abstraction.

Identifying objects

Consider the integer 2, the real number 2, and the complex number 2.

Set theory tells us $\mathbb{Z} \subseteq \mathbb{R} \subseteq \mathbb{C}$.

By there is also a 2 in the p-adics for each p and a 2 in the quaternions. Are these the same 2?

Identifying objects

In fact, there is a 2 in every ring.

Identifying objects

In fact, there is a 2 in every ring.

The 2 in $\mathbb{Z}/2\mathbb{Z}$ is 0.

That *can't* be the integer 2, so maybe restrict to characteristic 0.

Identifying objects

In fact, there is a 2 in every ring.

The 2 in $\mathbb{Z}/2\mathbb{Z}$ is 0.

That *can't* be the integer 2, so maybe restrict to characteristic 0.

The 2 in $M_2(\mathbb{R})$ is

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

Is this the same as the integer 2?

Identifying structures

We also identify structures.

Identifying structures

We also identify structures.

Consider the following argument.

- $\{0, 1\}$ with addition modulo 2 is the two-element group.

Identifying structures

We also identify structures.

Consider the following argument.

- $\{0, 1\}$ with addition modulo 2 is the two-element group.
- $\{1, -1\}$ with multiplication is the two-element group.

Identifying structures

We also identify structures.

Consider the following argument.

- $\{0, 1\}$ with addition modulo 2 is the two-element group.
- $\{1, -1\}$ with multiplication is the two-element group.
- The identity of the two element group is the identity of the two element group.

Identifying structures

We also identify structures.

Consider the following argument.

- $\{0, 1\}$ with addition modulo 2 is the two-element group.
- $\{1, -1\}$ with multiplication is the two-element group.
- The identity of the two element group is the identity of the two element group.
- $0 = 1$.

Identifying structures

We also identify structures.

Consider the following argument.

- $\{0, 1\}$ with addition modulo 2 is the two-element group.
- $\{1, -1\}$ with multiplication is the two-element group.
- The identity of the two element group is the identity of the two element group.
- $0 = 1$.

It's not that mysterious: we can treat things as being the same as long as we

- only talk about things that respect the sameness, and
- don't talk about the differences.

The structural view of mathematics

When we talk about mathematical objects, we are talking about elements of structures.

The slogan: mathematical objects are *places* in structures.

We only care about structures up to isomorphism.

The structural view of mathematics

When we talk about mathematical objects, we are talking about elements of structures.

The slogan: mathematical objects are *places* in structures.

We only care about structures up to isomorphism.

But isomorphism requires having a particular *signature* in mind.

What counts as the signature depends on context.

The structural view of mathematics

When we talk about mathematical objects, we are talking about elements of structures.

The slogan: mathematical objects are *places* in structures.

We only care about structures up to isomorphism.

But isomorphism requires having a particular *signature* in mind.

What counts as the signature depends on context.

For example, what is i in the complex numbers?

The structural view of mathematics

When we talk about mathematical objects, we are talking about elements of structures.

The slogan: mathematical objects are *places* in structures.

We only care about structures up to isomorphism.

But isomorphism requires having a particular *signature* in mind.

What counts as the signature depends on context.

For example, what is i in the complex numbers?

When we talk about i in $\mathbb{C}$, we mean $\mathbb{C}$ with a fixed choice of i.

Managing abstraction

That's all well and good; but when we formalize mathematics, we have to operationalize that.

In other words, we need to track when something we have established in one context can be applied in another.

Strategies in formalization:

- Generalize.
- Use equality.
- Use algebra.
- Use coercions.

Strategy 1: Generalize

Try not to prove things about one concrete structure.

For example, prove $\forall x\,(x + y = y + x)$ generically.

Then it applies to $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and we don't have to worry about sameness.

Strategy 1: Generalize

Try not to prove things about one concrete structure.

For example, prove $\forall x\ (x + y = y + x)$ generically.

Then it applies to $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and we don't have to worry about sameness.

Mathematically, we often think of $\mathbb{R}^n$ as $\{0, \ldots, n-1\} \to \mathbb{R}$.

Then it becomes hard to identify $\mathbb{R}^{m+n}$ with $\mathbb{R}^m \times \mathbb{R}^n$.

Strategy 1: Generalize

Try not to prove things about one concrete structure.

For example, prove $\forall x\ (x + y = y + x)$ generically.

Then it applies to $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and we don't have to worry about sameness.

Mathematically, we often think of $\mathbb{R}^n$ as $\{0, \ldots, n-1\} \to \mathbb{R}$.

Then it becomes hard to identify $\mathbb{R}^{m+n}$ with $\mathbb{R}^m \times \mathbb{R}^n$.

In Lean, we find it convenient to let n be any finite indexing type.

Strategy 2: Use equality

Arranging $\mathbb{Z} \subseteq \mathbb{R} \subseteq \mathbb{C}$ can be convenient at times.

These are all the same as subsets (and hence subfields) of the reals:

- $\mathbb{Q}[\sqrt{2}][\sqrt{3}]$
- $\mathbb{Q}[\sqrt{3}][\sqrt{2}]$
- $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$

The *Mathematical components* library in Coq/SSReflect favors theorems about *subgroups* over theorems about groups, because then you don't have to identify elements with those of the ambient group.

Strategy 3: Use subgroups

Build isomorphisms and embeddings into all your theorems.

- There is an action of $\mathbb{Z}$ on any abelian group.
- Instead of saying R is a subring of S, prove things about R-algebras S.
- Equivalently, proving things about homomorphisms from R to S.
- Instead of linear spaces, reason about linear mappings.

(Baanen, Dahmen, Narayanan, and Nuccio describe the last strategy in a nice paper on class field theory in Lean.)

There is no need to identify elements if the theorems are about the identifications.

Strategy 4: Use coercions

Sometimes it is impossible to avoid identifying a structure with its isomorphic image.

Theorem provers support coercions, e.g. silently inserting a coercion from $\mathbb{Z}$ to $\mathbb{R}$.

It can be a pain in the neck. Automation helps.

Identification: an objection

Mathematicians identify structure all the time, effortlessly, without thinking about it.

I have described an ad-hoc list of coping mechanisms.

Surely there must be a simpler story as to what is going on?

Identification: a response

Learning to do mathematics is hard.

Learning to think mathematically requires learning how to mediate between different views of structures.

It seems likely that we internalize various strategies like these, without realizing it—to the point that, when pressed, it's hard to spell out the justification explicitly.

Let's take the phenomenon seriously, and be scientific about it.

Table of contents

Table of contents

Proofs

Proofs, in mathematics, are the coin of the realm.

In formal logic, a proof is sometimes a sequence of lines, where each subsequent line follows from previous ones by a rule of inference.

Sometimes logicians like to draw diagrams.

In real life, proofs are mostly linear.

Proof by cases is clearly marked.

A proof by induction involves a base case and an induction step; the first is usually dispensed with quickly.

Proofs

What's the biggest difference between informal and formal proofs?

Proofs

What's the biggest difference between informal and formal proofs?

Example: let p be prime, suppose $n > 0$, and suppose $p \mid n$. Then $n \geq 2$.

Proofs

What's the biggest difference between informal and formal proofs?

Example: let p be prime, suppose $n > 0$, and suppose $p \mid n$. Then $n \geq 2$.

```
example (p n : ℕ) (prime_p : prime p)
  (pos_n : n > 0) (p_div_n : p ∣ n) : n ≥ 2 :=
le_trans prime_p.two_le $ le_of_dvd pos_n p_div_n
```

Proofs

What's the biggest difference between informal and formal proofs?

Example: let p be prime, suppose $n > 0$, and suppose $p \mid n$. Then $n \geq 2$.

```
example (p n : ℕ) (prime_p : prime p)
  (pos_n : n > 0) (p_div_n : p ∣ n) : n ≥ 2 :=
le_trans prime_p.two_le $ le_of_dvd pos_n p_div_n
```

Informal proofs favor statements over reasons. Most reasons don't have names.

Proofs

Coping mechanisms in proof assistants:

- Use automation where possible (ranging from explicit automation to type class inference).
- Use naming schemes and tab completion.
- Use search engines.
- Use browser pages.
- Just learn the names of things in the damn library.

In informal mathematics, various devices are used to indicate reasons, implicitly or explicitly.

Proofs

"Hence the closure of S in E is $\bar{E} \cap S$."

"By our previous results the first case obtains when $p \equiv 1 \bmod 4$ and $p \equiv \pm 1 \bmod 12$."

"Because H is connected, the fiber F over $\gamma(0) = \gamma(1)$ is connected."

"Now we can see that all the maps f_* and $\bar{f}_*$ in the commutative diagram of transfer sequences are isomorphisms by induction on dimension, using the evident fact that if three maps in a commutative square are isomorphisms, so is the fourth."

Proofs

"From $P[Y \in D_f] = \mu(D_f) = 0$ it follows that $f(Y_n) \to f(Y)$ with probability 1, and so by change of variable (see (21.1)) and the bounded convergence theorem,
$\int f \, d\mu_n = E[f(Y_n)] \to E[f(Y)] \to \int f \, d\mu$."

"Using Hilbert's theorem 90 again, together with the fact that $|\sigma\alpha| = |\alpha|$ for all $\alpha \in K^*$, we see that $H = K^{*1-\sigma}$."

Proofs

Sometimes a justification

- . . . is an explicitly named theorem.
- . . . is a named property.
- . . . is a statement that has been numbered or starred.
- . . . involves repeating a fact previously established.
- . . . involves a few words of explanation.

It is often just left implicit.

Proofs

Sometimes justification is flagged in advance.

"Note that adjoining a copy of $\mathbb{Z}$ induces an injection on $\pi_1 \ldots$"

"Note that each set U'_n is open, being the difference of an open set U_n and a closed set $\bigcup_{i=1}^{n} \overline{V}_i$."

What sort of cognitive fact is *noting*?

Think about the other cognitive acts we are called on to perform.

Proofs

"Expressing each generator as a sum of homogeneous elements, . . . "

"We can choose the indices i in such a way that $\sigma_i A_1 = A_i$."

"Now identify $\mathfrak{g}$ with R^3 by means of an orthonormal basis with respect to the $\mathrm{Ad}\, G$-invariant inner product on $\mathfrak{g}$."

". . . consider the n simplex $[\nu_0, \cdots, \nu_n]$ with ν_0 at the origin and ν_k the unit vector along the kth coordinate axis for $k > 0$."

"We now apply the bilinear form of Parseval's identity (Lemma 1.5, Chapter 3) to the integral defining $\mathcal{A}$."

Proofs

We are called on to:

- express things
- choose things
- identify things
- consider things
- apply things

In each case, what are we called on to do?

How does this translate to formal terms?

Plans

Consider also the temporal, narrative devices.

"Now let $\mathfrak{c}$ be a connected component of $\mathfrak{t}^{\mathrm{reg}}$."

"To show that ψ is onto, it is sufficient to show that . . ."

"Having shown that ψ is onto, we now investigate the kernel. Clearly, $\ker \psi = A_1 \cap A_2 \cap \cdots \cap A_g$. We must show that under the hypotheses the intersection is equal to the kernel."

"Thus it only remains to show that (ii) $\Rightarrow$ (iii). To do this. . ."

Plans

"We are now in a position to state the main theorem of this chapter. The proof will be spread out over the next three sections."

"There would be no difficulty if we could write $B = A[\alpha]$ for some α. This is true only locally. Hence we shall use the approximation theorem to reduce our problem to the local case."

Plans

The philosopher of mathematics Rebecca Morris has explored the notion of what it means for a step in a proof to be *motivated*.

She and Yacin Hamami have tried to understand proofs in terms of *rational plans*, drawing on the philosophy of planning and agency.

We have mathematical languages to express:

- definitions
- theorems
- proofs

Why not languages to express mathematical plans?

Conclusions

Understanding the nuances of language used in proofs may take us beyond conventional formal logic.

On the other hand, it isn't like understanding literature and poetry.

There are common protocols, idioms, and strategies.

Classifying them is a study unto itself.

Conclusions

Understanding specifically mathematical uses of language is important for:

- formalizing mathematics
- supporting (and automating) mathematical inference
- learning from or mining mathematical texts

It's also interesting, and helps us appreciate the mathematics.

A good proof is a work of art. But it is also well designed.

Parting thought

Niklaus Wirth recently wrote (CACM 3/21):

"The principal purpose of a higher-level language is to raise the level of abstraction from that of machine instructions. Examples are data structures vs. word arrays in memory, or conditional and repetitive statements vs. jump instructions. A perfect language should be defined in terms of mathematical logic, of axioms and rules of inference. *No reference to any computer or mechanism should be necessary to understand it*. This is the basis of portability."

Note: we implement an algorithm in a programming language, which, in turn, must be implemented by a machine model.

Parting thought

Here is my take:

"The principal purpose of mathematical language is to raise the level of abstraction from that of formulas and axioms. Examples are sets and functions vs. their formal representations, or algebraic structures vs. their set-theoretic definitions. A perfect language should be defined in terms of mathematical ideas, of concepts and methods of reasoning. *No reference to any formal system should be necessary to understand it*. This is the basis of communication."

Idea: we implement a piece of mathematics in a mathematical language, which, in turn, must be implemented by a formal foundation.