The design of mathematical language

Jeremy Avigad

August 29, 2021

Abstract

As idealized descriptions of mathematical language, there is a sense in which formal systems specify too little, and there is a sense in which they specify too much. They are silent with respect to a number of features of mathematical language that are essential to the communicative and inferential goals of the subject, while many of these features are independent of a specific choice of foundation. This chapter begins to map out the design features of mathematical language without descending to the level of formal implementation, drawing on examples from the mathematical literature and insights from the design of computational proof assistants.

1 Introduction

Mathematics is governed by a network of norms that determines what we can say and how we are supposed to say it. Learning how to speak about mathematics properly is an important part of learning to do mathematics. But, when pressed, it turns out to be surprisingly hard to say what it is we are talking *about*; mathematics seems to be, as the title of a book by John Burgess and Gideon Rosen proclaims, a subject with no object. And without making sense of what we are talking about, it is hard to see how we can make sense of what we are thinking about. The outward manifestations of mathematical thought are found in the way we communicate mathematics to others.

Here we adopt the view that the best way to make sense of what we are talking about is to develop a better understanding of mathematical language itself. From a grammatical point of view, the language of mathematics is rather simple. There are no subtle variations of tense, modality, or aspect, and the subject is generally devoid of subjunctives and counterfactuals. Mathematical statements make claims as to what is true, always has been true, and always will be true, and mathematical proofs back these claims up with more statements of the same sort. The subtleties of mathematical language stem rather from the features that render it suitable for reasoning about abstract objects and properties.

Formal languages provide informative models of mathematical discourse. It is by now well understood that ordinary mathematics can be formalized in various ways in first-order logic and the language of set theory, in variants of simple type theory or dependent type theory, or in category-theoretic foundations. These variations provide complementary perspectives and serve as bases for formalization in proof assistants like Mizar [Grabowski et al., 2010], Isabelle [Nipkow et al., 2002], Coq [Bertot and Castéran, 2004], HOL Light [Harrison, 2009], Metamath [Megill, 2006], and Lean [de Moura et al., 2015].

As valuable as they are, there is a sense in which formal languages specify too much, and there is a sense in which they specify too little. The fact that an ordinary mathematical text can be represented equally well in any of the foundations above shows that these foundations should be viewed as alternative *implementations* of mathematics. It is therefore reasonable to look for descriptions of mathematical language and patterns of inference that abstract away the superficial differences and clarify the specifications that the implementations are designed to meet.

Formal languages specify too little in the sense that many essential features of ordinary mathematical language are not addressed by a formal specification. Instantiating the formal foundation is only the first step in implementing a mathematical proof assistant, and the bulk of the work then goes into supporting the interactions that make them usable in practice. Such systems have to parse user input, disambiguate notation, manage libraries of theorems and definitions, and keep track of algebraic structures and relationships between them. They need to provide convenient manners of expression and support efficient inference. The design of a proof assistant requires countless engineering decisions that bear on the system's usability, and we might optimistically seek a broad view of mathematical language that can help us evaluate the choices and assess their merits.

Another approach to thinking about mathematical language is to view it as a part of natural language, albeit a part of natural language with its own characteristic features. Ganesalingam [2013] leans in this direction, which allows him to bring the methods of generative linguistics to bear on the analysis of ordinary mathematical texts. Various systems of *controlled natural language* offer a complementary approach, presenting structured languages with enough flexibility to incorporate a range of natural language constructs, so that users can write texts that read like natural language but can be translated to the language of an underlying formal system [Paskevich, 2007, Cramer et al., 2009].

Here we will not try to account for the range of grammatical and stylistic variation that one finds in natural language. We will rather seek idealizations that illuminate the features of mathematical language that are specifically adapted to supporting mathematical activity. One way of describing what we are after is to say that we are looking for a description of mathematics as a *semiformal language*. We want a description that renders it regimented and precise, like a formal language, designed to support certain types of abstraction and inference. At the same time, we want a model that is more informative than formal logic, one that tells a story of how mathematical language serves to support the processes that are needed to track mathematical objects and the relationships between them. Our overarching goal will be to develop such a design specification for mathematical language without descending to the level of a fully formal implementation.

This chapter falls short of that goal, but it aims to make progress by cataloging some of the features of mathematical language that are essential to its communicative and inferential aims. It also describes some of the mechanisms that contemporary proof assistants use to model these features and support these aims. The interaction between informal and formal mathematics thus gives rise to a fruitful dialectic: formalizing contemporary mathematics helps clarify the requirements that a good proof assistant must meet, and engineering efforts to satisfy those requirements help clarify the nature of mathematical language and inference.

2 Perspectives

2.1 Philosophical orientation

Philosophy of mathematics has traditionally been concerned with the nature of mathematical objects, knowledge, and thought. Talking about these in a rigorous way requires some sort of conception of mathematics itself, and some sort of understanding the features of mathematical practice that fall under the scope of the analysis. Toward forming such a conception, what we have the most direct access to is the mathematical literature: the historical record of statements, questions, arguments, definitions, and and other textual artifacts that are constitutive of the subject. These artifacts, rooted in language, form the starting point for philosophical study.

However we ultimately try to characterize the goals and methods of mathematics, we have to start with language. Whether we view mathematics as the practice of solving problems, abstracting from experience, or getting at a certain type of truth, what we say about that practice has to fit with what we see in the mathematical literature. We need to understand how mathematical language enables us to carry out those activities and how those activities are manifested in language.

One central thesis of this chapter is that, when we study mathematical language, it is important to understand not just what is *allowed*, but also what is *desired*. Formal systems specify rules that tell us when a formula is well formed and when an inference is justified, but it doesn't tell us which definitions are good definitions, or which among the myriad inferential steps that can be taken at any given point are most worthy of our attention. Mathematics calls upon its practitioners to carry out complex tasks, and to do so creatively, efficiently, and reliably. Reflection on mathematical practice should help explain how it helps us manage complexity and carry out fruitful exploration.

Another central thesis of this chapter is that it is helpful to view mathematical language as the object of *design*. Mathematical language and method have evolved over the centuries, presumably for good reasons. Some features of mathematical language and method have remained remarkably stable, again, we may presume, for good reasons. Mathematics provides powerful means for abstraction and for managing information, making data salient when it is needed and suppressing it when it is a distraction. Recognizing that it is effective in that regard because it has been designed for that purpose encourages us to uncover the design principles at play.

2.2 Methodological orientation

With respect to the goal of developing a semiformal description of mathematical language, and how can tell whether we are on the right track? This chapter uses two sources of data as touchstones.

The first is the mathematical literature itself. In the sections that follow, I will frequently resort to examples pulled from a random sample of textbooks. There is nothing exceptional about the examples I chose; the point is that they represent patterns that one can find in virtually any mathematical text. A theory of mathematical language should help us make sense of the instances of mathematics that we find all around us.

The second source of data comes from contemporary proof assistants, and from the formal methods that have been developed to support mathematical communication and inference in that setting. Processing a mathematical text requires tokenizing, parsing, and elaboration, and a variety of methods are used to infer information that is generally left implicit, including class inference, canonical instances, and unification hints [Wenzel, 1997, Sozeau and Oury, 2008, Asperti et al., 2009, Garillot et al., 2009]. Theories and libraries are organized using environments, modules, and namespaces. Decision procedures support domain-specific reasoning, and indexing techniques provide fast ways of locating relevant information for automated search [Robinson and Voronkov, 2001]. Ongoing research is bringing the methods of machine learning to bear on mathematical reasoning [Gauthier et al., 2021, Lee et al., 2020].

Describing this technology in detail would take us too far afield. (For overviews, see Hales [2008], Avigad and Harrison [2014], Avigad [2018], Blanchette and Mahboubi [to appear].) The technological developments are informative, however, because they make explicit the challenges that need to be met in order to bridge the gap between informal mathematics and its formal representation. A second measure of success for the analysis here, then, is the extent to which it helps clarify the technical challenges.

We will focus on contemporary mathematical language. It is also worthwhile to study the way mathematical language has changed over time and to try to understand the reasons behind those changes, but I will make no attempt to do that here. When I refer to historical sources in this chapter, it is either to highlight features of mathematical language that have remained fairly stable over time or to highlight features of mathematical language that are relatively new. In both cases, the aim is to characterize mathematical language as it is used today.

3 Fundamentals

3.1 Sorts

We start with the simple observation that mathematical objects fall into categories. Consider the first proposition of Euclid's *Elements*:

> 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.
>
> [Euclid, 2002, p. 3]

The proof refers to a number of different kinds of objects: points A, B, and C, a finite straight line AB, a circle BCD, and the equilateral triangle that is under construction. Some objects arise in the course of the proof: given the point A and the finite straight line AB (now construed as a distance), a circle is described, and given the points C and A, the straight line CA is joined. Objects bear relationships to one another: two circles can cut one another at a point C, and an equilateral triangle can lie on a straight line.

It is informative to consider the way that objects are introduced into contemporary mathematical discourse.

> Let X be a locally convex topological vector space and F a closed convex subset.
> [Royden, 1988, p. 241]

> Let p and q be odd primes.
> [Ireland and Rosen, 1990, p. 53]

> 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.
> [Lang, 2002, p. 378]

> Let $\mathfrak{M}$ be a maximal ideal of $k[x]$. Let σ be the canonical homomorphism $\sigma : k[x] \to k[x]/\mathfrak{M}$.
> [Lang, 2002, p. 378]

> Let $\tau_2 = n$; let t_1 be the smallest k such that $X_k \geq \alpha$, if there is one, and n otherwise.
> [Billingsley, 1995, p. 5]

The first three examples appear in the statements of theorems, whereas the next two are taken from the beginnings of proofs. The point is simply that we would not expect to see a theorem or proof begin with the words "let p and q be objects" or "let p and q be things." Mathematical variables always range over a *sort*.

It is not always clear what we should take the fundamental sort of an object to be, and what we should view as ancillary attributes. It seems sensible to say that an equilateral triangle is fundamentally a triangle that happens to have the property of being equilateral, and it seems sensible to say that a prime number is a number that is prime. But is a nonnegative integer and integer that has the property of being nonnegative, or a number that has the property of being nonnegative and integral? Is a finite straight line in the *Elements* a straight line that happens to be finite? A maximal ideal of $k[x]$ can be construed as an ideal of $k[x]$ that has the property of being maximal, or a subset of $k[x]$ that has the property of being a maximal ideal, or simply a set that has the property of being a maximal ideal of $k[x]$. In axiomatic set theory, everything is a set, but that stance is not more informative than considering everything an object.

The fact that objects belong to sorts is not specific to mathematics. Aristotle distinguished between an object's *essential* and *accidental* properties. Socrates was both human and short, and we can imagine what ancient Greece might have been like had Socrates been a few inches taller, but when we try to imagine Socrates as anything other than a human being, it's hard to argue that we are still thinking about Socrates. The ascription of sorts to mathematical objects, however, has specifically mathematical uses.

- It can be used to disambiguate notation. The expression x^y might denote exponentiation, but it might also denote conjugation in a group. The expression $x \cdot y$ can denote multiplication in any structure, so its meaning depends on the sort of objects that x and y are.
- It can be used to determine meaning. An element x can be *maximal* only if it is viewed as an element of a structure with an associated ordering. A function f is *surjective* only with respect to a specification of its codomain, *continuous* only with respect to a topology on its domain and codomain, a *homomorphism* only with respect to algebraic structures on those, and *essentially bounded* only with respect to a measure on the domain and an ordering of the codomain.

- It can be used to justify an expression as meaningful; for example, $\sqrt{x}$ makes sense if x is a nonnegative real number, and $1/x$ is meaningful if x is a nonzero real.
- It can be used to support immediate inference. If $f : G \to H$ and $g : H \to K$ are group homomorphisms, their composition is again a group homomorphism. If we view them fundamentally as functions, then their composition is merely another function, and we need an additional inference to conclude that they are again homomorphisms.
- It can be used to support higher-level inferences and heuristics. Knowing that we are dealing with integers or real numbers or triangles or elements of a group cues us to a body of methods and background knowledge that are relevant to the inferential tasks at hand.

We can designate the sorts of objects that occur in the passage from Euclid with tokens like Point, Line, and Circle. In the theory of programming languages and in a number of computational proof languages, it is common to view these as *data types*, or, more simply, *types*. We can write $A : \mathsf{Point}$ to express that a variable or expression A is a point and write $L, M : \mathsf{Line}$ to express that L and M are lines.

In formal axiomatic foundations like *simple type theory* and *dependent type theory*, every expression denotes an object of some type. Proof assistants based on those foundations therefore have built-in mechanisms for *type inference*, which is the task of determining the type of a given expression, and *type checking*, which is the task of checking whether the type of an expression matches an expected type. The latter can be used to report errors when users apply operations to the wrong sorts of arguments, or when the system cannot infer the information it needs to make sense of an expression. Even systems based on set theory sometimes incorporate notions of type. *Soft typing* refers to the use of a typing discipline separate from the axiomatic foundation to infer meaning and to report errors.

3.2 Operations

Mathematical operations provide constructions or descriptions of objects. Two points A and B determine a line segment AB and, in the excerpt from the *Elements*, ACE is specified as the circle with center B and radius equal to the distance AB. The first operation may be modeled with a constant $\mathsf{segment\text{-}of} : \mathsf{Point} \to \mathsf{Point} \to \mathsf{Segment}$, where the type indicates that $\mathsf{segment\text{-}of}$ can be applied to two points to yield a segment. The convention in proof assistants is to take the arrows to associate to the right, so that the type above is read $\mathsf{Point} \to (\mathsf{Point} \to \mathsf{Segment})$. This means that when $\mathsf{segment\text{-}of}$ is applied to a point A, the result $\mathsf{segment\text{-}of}\ A$ has type $\mathsf{Point} \to \mathsf{Segment}$. When this is applied to another point B, the resulting expression $\mathsf{segment\text{-}of}\ A\ B$ has type $\mathsf{Segment}$. The constant $\mathsf{segment\text{-}of}$ in $\mathsf{segment\text{-}of}\ A$ is said to be *partially applied*, and considering such expressions is natural in some mathematical contexts but not others.

The second operation may be modeled by a constant $\mathsf{circle\text{-}of} : \mathsf{Point} \to \mathsf{Segment} \to \mathsf{Circle}$, or perhaps $\mathsf{circle\text{-}of} : \mathsf{Point} \to \mathsf{Point} \to \mathsf{Circle}$ if it turns out that the distance in question is always measured from the center. Arithmetic offers other prototypical examples of mathematical operations: we can refer to the sum of two numbers, $x + y$, the square x^2 of x, or the greatest common divisor $\mathsf{gcd}\ x\ y$ of x and y.

In most formal foundations, a function f has an intended domain A and an intended codomain B, which is what is expressed by writing $f : A \to B$. In ordinary mathematics, it is well understood is that the interpretation of the codomain is fluid: the specification $f(x) = x^2$ can be viewed equally well as defining a function $f : \mathbb{R} \to \mathbb{R}$ and as defining a function $f : \mathbb{R} \to \mathbb{R}^{\geq 0}$, where $\mathbb{R}^{\geq 0}$ denotes the nonnegative real numbers. More confusingly, the interpretation of the domain is also fluid: in a mathematical context, $f(x) = x^2$ may be viewed as a function defined on the complex numbers, real numbers, rationals, integers, or natural numbers, or any other domain with a multiplication. These considerations bear on whether we take the squaring function to be injective, surjective, both, or neither. In mathematical contexts there is generally a broadest construal of the domain that makes sense, and circumstances dictate when we need to pass to a restricted interpretation.

The fact that the dot in $x \cdot y$ can denote multiplication in any of a number of structures and that an expression $\mathsf{intersection}\ X\ Y$ may denote the intersection of two lines, two circles, or a line and a circle shows that mathematical notation and terminology are *overloaded*. In these cases, information about the sorts of the arguments can be used to disambiguate meaning. In some cases, it is natural to view information like the structure in which multiplication occurs as an *implicit argument* to the notation. Proof assistants invariably have mechanisms to infer such information. A semiformal specification of mathematical language should say something general about how ambiguities are resolved and how and when implicit information is inferred.

3.3 Predicates and relations

If we use $\mathsf{center\text{-}of}$ to denote the relation of being the center of a circle, computer scientists are apt to write $\mathsf{center\text{-}of}\ A\ \gamma$ to express that A is the center of γ. In a proof assistant we might write

$$\mathsf{center\text{-}of} : \mathsf{Point} \to \mathsf{Circle} \to \mathsf{Prop}$$

to specify the type of the the relation. We continue to interpret the arrows as associating to the right, so that $\mathsf{center\text{-}of}\ A\ \gamma$ denotes a proposition, namely, the proposition that A is the center of γ. Modeled in this way, relations can take any number of arguments. A relation that takes only one argument is usually called a *predicate* or *property*. For example, $\mathsf{equilateral}\ T$ expresses that a triangle T is equilateral and $\mathsf{even}\ n$ expresses that an integer n is even.

Notation and terminology for relations, like notation and terminology for operations, can also be overloaded. We can use a symbol $\mathsf{on} : \mathsf{Point} \to \mathsf{Line} \to \mathsf{Prop}$ to express that a point A lies on a line L, and a symbol $\mathsf{on} : \mathsf{Point} \to \mathsf{Circle} \to \mathsf{Prop}$ to express that a point A lies on a circle γ. Similarly, $x \leq y$ can be used denote an order comparison in any structure that has such an order. And we have already seen that predicates can rely on implicit information. A predicate $\mathsf{maximal}\ x$ depends on the order structure with respect to which x is judged to be maximal, and $\mathsf{continuous}\ f$ depends on the topologies with respect to which f is judged to be continuous.

3.4 Definedness and partiality

Another complication that arises in the formalization of mathematics is that mathematical language often uses operations that can only meaningfully be applied to some of the objects in the natural domain of interpretation. Euclid can only speak of the intersection of two circles when the two circles intersect, and the phrase "the greatest common divisor of x and y" only makes sense when x and y have a greatest common divisor, which is not the case when x and y are zero.

Set theory allows us to view each of these as a partial function, whose value may or may not be defined at a given input. Formally, a partial function from A to a B is a function from some subset of A to B. Proof assistants sometimes provide mechanisms for defining *subtypes* or adding *preconditions*, namely, proof obligations that must be fulfilled every time such a function is applied. For example, one can insist that every inscription a/b is accompanied by a proof that b is nonzero.

In ordinary mathematics, however, the implicit preconditions can be subtle. Consider a statement of the prime number theorem:

> Let $\pi(x)$ be the number of primes between 1 and x...
>
> $$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln(x)} = 1.$$
>
> [Ireland and Rosen, 1990, p. 2]

The expression in the limit is undefined when x is negative, equal to 0, or equal to 1. None of this matters because to determine the limit we can restrict attention to arbitrarily large values of x, but in a formal foundation, the range of relevant values and the interpretation of the expression at those values have to be made explicit. Similarly, consider the statement of Bézout's theorem in an introductory number theory textbook:

> ... Thus the greatest common divisor (b,c) is defined for every pair of integers b,c except for $b = 0, c = 0$, and we note that $(b,c) \geq 1$.
>
> **Theorem 1.3.** If g is the greatest common divisor of b and c, then there exist integers x_0 and y_0 such that $g = (b,c) = bx_0 + cy_0$.
>
> [Niven and Zuckerman, 1980, p. 7]

Here the phrase "the greatest common divisor" is rendered meaningful by an implicit assumption that b and c are not both zero.

One always has the option of modeling a partial operation as an operation on a smaller sort. For example, one can take the variable x in the statement of the prime number theorem as ranging over positive real numbers. This often leads to problems; we should be able to supply a positive real number in any context where a real number is expected. (See the discussion of *identification* in Section 4 below.) But in other contexts it is more natural. The product $M \cdot N$ of two matrices makes sense only if the number of columns of M is equal to the number of columns of N; in other words, if M is an $m \times k$ matrix and N is a $k \times n$ matrix for some m, k, and n. One can interpret multiplication as a partial operation on matrices, but formalization is often smoother if we interpret it as a family of total operations indexed by m, k, and n, and take these values to be implicit in the expression. This requires that a variable like M can range over a sort $\mathsf{Matrix}\ m\ k$ of $m \times k$ matrices, or even $\mathsf{Matrix}\ R\ m\ k$, where R specifies a ring of underlying elements. Such a sort is said to be a *dependent type*. The fact that $\mathsf{Matrix}\ R\ n\ n$ can be viewed as a ring for every $n > 0$ provides additional support for viewing $\mathsf{Matrix}\ R\ m\ k$ as an independent sort of objects rather than a predicate on a larger one.

4 Abstract objects

Mathematics is the art of rigorous abstraction. Mathematization amounts to identifying the features that are essential to the inferential and calculational steps that are needed in specific reasoning contexts and making them explicit. This results in *modularization* and *encapsulation*: the objects and assumptions that an argument depends are presented as part of the interface, and the argument can then be applied to any data that instantiates the relevant hypotheses.

Abstracting and modularizing in this way serves two purposes [Avigad, 2020]. The first is to manage complexity. If an argument about a geometric object only depends on the fact that it is a triangle, presenting it as such means we can temporarily ignore other information we have about it, such as its particular dimensions or its relationship to other objects in a diagram. If an argument about the roots of an equation only depends on the fact that the set of permutations of those roots is a group, presenting the argument in group-theoretic terms allows us to set aside everything else we know about equations and roots. If an argument about a complicated space of functions depends only on the fact that it is a convex Banach space, presenting it as such allows us to suppress specific details of the space that are irrelevant to the argument.

The second purpose is, of course, generality. A theorem about triangles holds of all triangles, not just the one in a particular diagrammatic configuration. A group-theoretic result holds of all groups, of which there are many, and a theorem about convex Banach spaces holds of every convex Banach space. An abstract argument can be applied not only in settings we are currently interested in, but also in settings we cannot yet even imagine.

In Sections 6 and 7, we will consider the way that set-theoretic language and the use of algebraic structures, respectively, support these goals. But first it is helpful to think about the way we talk about abstract objects in general, and the role that abstract objects play in mathematical thought.

4.1 Numbers

For centuries, mathematics was understood to be the general science of *quantity*, with continuous quantities, or *magnitudes*, as the subject matter of geometry, and discrete quantities, or *number*, as the subject matter of arithmetic. The first mention of magnitudes occurs in Book V of Euclid's *Elements*, which introduces the theory of ratios:

> A magnitude is *part* of a magnitude, the less of the greater, when it measures the greater.
>
> The greater is a *multiple* of the less when it is measured by the less.
>
> [Euclid, 2002]

In the *Elements*, two magnitudes may be equal or one may be less than another. A magnitude a is a *part* of a magnitude b if b can be *measured* as some whole number multiple of a. Book VII extends the terminology to *number*:

> An *unit* is that by virtue of which each of the things exist is called one.
>
> A *number* is a multitude composed of units.
>
> A number is *a part* of a number, the less of the greater, when it measures the greater.
>
> [Euclid, 2002]

Today, we are apt to interpret Euclid's magnitudes as nonnegative real numbers and his numbers as natural numbers. But Euclid's presentation and the early history of mathematics bring some interesting differences to light. One curious feature of the *Elements* is that the smallest number is 2; one sheep does not make a multitude. This fact is manifest in the first two propositions of Book VII, which describe the Euclidean algorithm for calculating the greatest common divisor of two numbers. The first proposition deals with the case where the result is a unit, whereas the second deals with the case where the two numbers have a common measure.

A more significant difference is that, in Euclid, a magnitude is always a magnitude of some type. It makes sense to add and compare lengths, areas, and angles, but not to add a length to an area or compare a length to an area. Even towards the end of the sixteenth century, we find François Viète, in the *New Algebra*, insisting that only homogenous quantities can be compared or added. So if x is a length, instead of $x^3 + x$, we must consider $x^3 + ax$ for some area a. Descartes' landmark *Geometry* of 1637 resolved the problem of comparing lengths and areas by providing a geometric construction that reduces products to lengths, modulo a choice of a unit.

Although today we consider numbers to be relatively concrete mathematical objects, we should keep in mind that they nonetheless represent a substantial abstraction. Mathematics allows us to add the average temperature in Pittsburgh on April 19, 2020 to the number of moons of Jupiter, and then multiply the result by the length of the Nile river, whether or not the resulting number is useful or interesting. The fact that we consider such a quantity to be a legitimate mathematical object says something significant about the nature of mathematical abstraction.

4.2 Dedekind abstraction

A parable by Paul Benacerraf imagines two children who, thanks to a solid logical upbringing, each possess a set-theoretic definition of the natural numbers [Benacerraf, 1965]. The definitions serve each of the children well, until they come to realize that although the definitions denote isomorphic set-theoretic structures, they are not identical. Since every object of set theory is, fundamentally, a set, mathematical objects have ancillary properties qua sethood. So we can imagine that according to one definition, 2 is an element of 3, and according to the other, it is not. Since the structures are isomorphic, the two students are in a position to come to agreement regarding all substantive claims about the set of natural numbers. But the identification of abstract numbers with concrete set representations forces us to the unsettling conclusion that, if the natural numbers really are sets, then either 2 is an element of 3 or it isn't, and any definition we give has to adjudicate the matter.

The title of Benacerraf's paper is a nod to an essay by Richard Dedekind from 1888, titled *Was sind und was sollen die Zahlen?* (roughly, what are numbers and what should they be?). That essay, and Dedekind's 1871 construction of the real numbers, foreshadow the problem. Dedekind explicitly addressed the issue in a letter to Heinrich Weber in 1888, in which he resists identification of the real numbers with his particular construction as a system of *cuts*:

> . . . I would advise that by [real] number. . . one understand not the class itself. . . but something new (corresponding to this class) which the mind creates. We are a divine race and undoubtedly possess creative power, not merely in material things (railways, telegraphs) but especially in things of the mind. This is precisely the same question that you raise at the end of your letter in connection with my theory of irrationals, where you say that the irrational number is nothing other than the cut itself, while I prefer to create something new (different from the cut) that corresponds to the cut and of which I say that it brings forth, creates the cut. . . . The rational numbers also produce cuts, but I would certainly not call the rational number identical with the cut it produces; and after the introduction of the irrational numbers one will often speak of cut-phenomena with such expressions, and ascribe to them such attributes, as would sound in the highest degree peculiar were they to be applied to the numbers themselves. Something quite similar holds for the definition of cardinal

number ... as a class; one will say many things about the class (e.g. that it is a system of infinitely many elements, namely, of all similar systems) that one would apply to the number only with the greatest reluctance; does anybody think, or won't he gladly forget, that the number four is a system of infinitely many elements? [Dedekind, 1888]

These circumstances have led a number of philosophers to adopt a *structural* view of mathematics (Resnik [1997], Shapiro [1997], Reck [2003], Parsons [2004]), which holds that when we say something about the number two, we are never talking about a unique object, but, rather, an element of an axiomatically characterized natural-number structure. Some go so far as to give the notion of a place in a structure a metaphysical standing of its own. The more sober mathematical practice, inaugurated by Dedekind, is to characterize mathematical structures uniquely up to isomorphism, provide a specific mathematical construction that shows that such a structure exists, and then refer to the structure only in terms of its axiomatic characterization, so that anything one says about the structure holds equally well of any isomorphic copy.

Perhaps there is no harm in identifying mathematical objects like numbers with particular set-theoretic objects if one the subsequently restricts oneself to the proper mathematical statements about them, but we should recognize that mathematical language obeys this restriction. Proof assistants implement various mechanisms to support this. In a number of proof assistants, once one has a construction of the real numbers as Dedekind cuts (or equivalence classes of sequences, or whatever), one can introduce a new type, $\mathbb{R}$, with an abstraction function abs from the type of representatives to $\mathbb{R}$, and a representation function repr in the other direction. These are assumed to satisfy $\mathsf{abs}\ (\mathsf{repr}\ y) = y$. The type $\mathbb{R}$ provides the "new things" suggested by Dedekind. The function $\mathsf{abs}\, x$ maps every representation to the abstract thing it represents, and the axiom says that every real number has a representation, not necessarily unique.

Inductive types in a proof assistant allow one to declare the natural numbers axiomatically as a type that is freely generated by zero and a successor function. Such declarations are underwritten by a set-theoretic interpretation, but the form of the declaration in the system helpfully abstract away the specifics. Alternatively, some systems, like Agda and Coq, offer systems of *modules*, which allow users to declare constants and properties axiomatically, reason about them, and instantiate them later on.

4.3 Equality

We will see in the next two subsections that mathematics allows us to substitute one expression for another in contexts that respect the sense in which we can treat them as the same thing. The equality relation in mathematics is generally reserved for substitutions that are allowed in all mathematical contexts. Saying $a = b$ means that we should be able to substitute a for b in any meaningful mathematical statement *salva veritate*, that is, preserving the truth of that statement. Frege's *On sense and reference* provides examples of linguistic contexts, like ascriptions of knowledge or belief, where identity and substitutability come apart. One of the hallmarks of of mathematical language is that it is devoid of such modalities.

If we assume that any meaningful mathematical statement about an object x describes a property of x, the requirement above follows from the principle that if two arguments are equal, they have all the same properties. Conversely, if two objects have all the same properties in common, then they must be equal: being equal to x is a property of x, so if y has all the same properties that x does, it is equal to x as well. This characterization of equality is known as the *Leibniz principle* or the *principle of identity of indiscernibles*.

Leibniz' principle says something about equality in general but but nothing about what it means for two objects to be equal in particular. It is perfectly reasonable to ask when two natural numbers are equal to each other; the inductive characterization of the natural numbers provides an answer to that, namely, that they are equal if they are both zero or have equal predecessors. Similarly, if we define the real numbers as equivalence classes of Cauchy sequences, asserting the equality of two real numbers is tantamount to saying that the Cauchy sequences representing them are equivalent. Questions of equality involve representations of objects, and settling such questions can require substantial mathematical knowledge.

We have seen that foundational approaches often overspecify mathematical objects. In set theory, we can ask whether $1 + 1$ is equal to the set $\{\emptyset, \{\emptyset\}\}$, a mathematical variant of Frege's Julius Ceasar problem. Dedekind abstraction helps; in most implementations of type theory, a proposition $s = t$ is grammatically acceptable only when s and t are objects of the same type.

One thing that plagues proof assistants is that, in addition to substantial judgments of equality, there are also implicit equality judgments that are so fundamental that they have a silent bearing on the very grammar of the language. For example, if x and y are column vectors of length 3, then concatenating them results in a vector of length 6. In a mathematical text, one might use

$$M \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

to denote the product of an $n \times 6$ array with that vector. Strictly speaking, the concatenation of x and y yields a vector of length $3 + 3$, and a tiny bit of mental arithmetic is needed to see that the expression is well-formed. Accepting this fact opens the door to allowing any background mathematical knowledge to bear on such determinations, and it isn't clear where to draw the line. Systems of type theory that allow ordinary mathematical expressions to occur in an expressions datatype (like $\mathbb{R}^{3+3}$ and $\mathbb{R}^6$) distinguish between *definitional equality*, which are the kind that the system needs to recognize to in order to judge whether an expression is well formed, and *propositional equality*, which is the more substantive kind. Informal mathematics does not make the distinction, and we still need to better understand how to model the identifications that mathematicians are not even conscious of performing.

4.4 Identifying objects

If there is any such thing as a concrete mathematical object, the number two should be one. But we have seen that even the number two is a rather abstract thing, and its nature is slippery. There are a 2 in the integers and a 2 in the rationals, and also 2s in the real numbers and complex numbers. Set theory tells us that we have $\mathbb{Z} \subseteq \mathbb{R} \subseteq \mathbb{C}$, so we can view these all as the same 2. But such a view is hard to maintain as the 2s proliferate. There are also 2s in the quaternions and octonians, and there is a 2 in the ring of p-adic integers for every prime number p. In fact, there is a 2 in every ring. In $\mathbb{Z}/2\mathbb{Z}$, we have $2 = 0$, so that particular 2 probably isn't the same as the integer 2. Even if we restrict attention to rings of characteristic zero, which contain an isomorphic copy of $\mathbb{Z}$, it is hard to maintain that all the 2s are the same. The 2 in the ring $M_2(\mathbb{R})$ of 2×2 matrices over $\mathbb{R}$ is

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

and it is hard to see how that can be equal to the integer 2.

The general mathematical stance is that what we are really doing is *identifying* all the isomorphic copies of the integers, allowing us to transfer properties between them. In particular, identifying the integers with a subset of the real numbers means that we can consider an object n to be an integer and a real number at the same time. So, in an expression like $3x^2 + 2x + 7$ we can view the coefficients as integers, we can view x as a real number, and we can view the multiplication as the usual multiplication operation on the real numbers. But each number domain has its own special character. We can prove a general expression involving a natural number n by induction on n only if we view n as a natural number and not a real number, and we can use the implication from $n < m + 1$ to $n \leq m$ when we know m and n are integers. When we write the binomial theorem,

$$(x+y)^n = \sum_{i \leq n} \binom{n}{i} x^i y^{n-i},$$

we understand that i ranges over natural numbers, or, equivalently, nonnegative integers, x and y are elements of any commutative ring, and the apparent multiplication by $\binom{n}{i}$ is interpreted as an action of integers on ring elements. In fact, in the identity

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

we recognize that the division symbol is justified a priori because any integer can be viewed as a rational number, though at the same time we are aware of the fact that in this case the operation keeps us within the integers.

The fluidity with which we pass between views of mathematical objects poses challenges for formalization. We will consider some of the practical mechanisms used to cope with this in Subsection 4.6.

4.5 Identifying structures

We also identify structures. Any mathematician will tell you that the set $\{0, 1\}$ with addition modulo two is the two-element group, and that the set $\{1, -1\}$ with multiplication is also the two-element group. The reflexivity of equality tells us that the identity element of the two-element group is equal to the identity element of the two-element group. By substitution, we can conclude that 0 is equal to 1.

As a mathematical joke, this isn't a very good one. There is nothing mysterious going on here: mathematicians are well aware of the fact that we can treat distinct objects as being the same only insofar as we limit ourselves to operations and properties that respect the senses in which they are the same and avoid talking about the differences. The two structures in the last paragraph are the same only insofar as we talk about them as groups. If we allow ourselves to talk about the elements of the carrier set, they are patently different.

In logical terms, it is permissible to identify isomorphic structures in the sense that any statement that can be expressed in the language of the structure's signature cannot distinguish between them. But then it is important to recognize that the relevant signature depends on the reasoning context at hand. Keränen [2001] challenged the philosophical view that mathematical objects are merely places in structures with the observation that some structures have nontrivial automorphisms. If the complex number i is just a place in a structure, which one is it? Anything that can be said about i can be said equally well about $-i$, so there is nothing to distinguish the two.

Again, as a mathematical joke, this isn't a very good one. When we talk about the complex numbers, we know full well that there are two square roots of -1, and that i can be designated to be either one. So the best way to construe talk about the complex numbers while viewing it as an algebraically characterized structure is to simply view it as a structure with a fixed choice of i. With that choice, the structure is rigid, and the problem is solved.

4.6 Managing abstraction

The conclusion of the previous subsection is that it is safe to identify structures as long as suitable linguistic hygiene is maintained. But mathematicians don't go around declaring languages and signatures; in any given context, they simply know what is allowed. When we formalize mathematics with a proof assistant, we have to operationalize that. It's not enough to make vague gestures at reasoning contexts; the rules need to be spelled out in enough detail that a proof assistant can supply the relevant axiomatic justification. We also need to make sure the different contexts interact, so that we can establish mathematical claims once and for all and then apply them wherever it is appropriate to do so.

In this subsection, I will describe four general strategies that are used in formal libraries to support the application of abstract theorems in the various contexts in which they can to be applied. The four strategies are as follows:

1. Generalize.
2. Use equality.
3. Use algebra.
4. Use coercions.

The first strategy can also be phrased in negative terms, in which case it sounds like cynical advice to a young politician: avoid saying anything specific. For instance, proof assistants usually establish $x + y = y + x$ as a general property of a suitable class of algebraic structures. That means that when we need to use the fact that $2 + 3$ is equal to $3 + 2$, we don't have to worry about whether this is a fact about integers or real numbers, or even whether they are the same fact or not. Nor do we have to worry about transferring the result from one domain to another. We just need to know that the relevant algebraic hypotheses are satisfied.

For that reason, mathematical libraries tend to favor theorems about commutative rings and ordered fields over theorems about specific number structures. Other opportunities for generality often crop up in unexpected ways. Theorems of multivariate analysis holds of the structure $\mathbb{R}^n$ of n-tuples of natural numbers, which we can view as maps from the set $\{0, 1, 2, \ldots, n-1\}$ to $\mathbb{R}$. But this can lead to the annoying need to reindex when we try to view $\mathbb{R}^n$ as an subspace of $\mathbb{R}^{m+n}$. In fact, most theorems one might want to prove about $\mathbb{R}^n$ remain true if we interpret n as an arbitrary finite set, and we can interpret the + in $\mathbb{R}^{m+n}$ as disjoint union. This avoids the need to treat any particular n-element set as canonical. If we incorporate this design decision into a formal library, we don't need to identify one n-element set with another because our theorems hold for all of them.

Nonetheless, there are invariably times when we want to consider a mathematical structure as both an entity in its own right and a substructure of a larger one. It is often convenient to have the integers be a subset of the reals. For a more striking example, consider the number field obtained by adjoining both $\sqrt{2}$ and $\sqrt{3}$ to the rationals. We can describe this structure as $\mathbb{Q}[\sqrt{2}][\sqrt{3}]$, $\mathbb{Q}[\sqrt{3}][\sqrt{2}]$, or $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$. These describe distinct mathematical constructions, any of which may come up in practice, leaving us struggling to identify them when the contexts clash. But viewed as subsets of the real numbers, they are the same subset, and any element of one is an element of the other.

A variation on this second strategy played a fundamental role in the celebrated formalization of the Feit–Thompson theorem [Gonthier et al., 2013]. The formalization of group theory used for that project might more accurately be called a formalization of *subgroup* theory because theorems in the underlying library where phrased as theorems about subgroups of a larger ambient group. Every group is a subgroup of itself, so, mathematically, there is no difference. But an element of a group can generally be viewed as an element of multiple subgroups of interest, and centering the library on the concept of a subgroup avoids the need to mediate between the different views.

The third strategy in the list is really a special case of the first: sometimes the right way to generalize a theorem involves adopting a proper algebraic perspective. Mathematicians are well aware of the fact that any abelian group A can be viewed, equivalently, as a $\mathbb{Z}$-module, which means that it comes equipped with an action $n \cdot x$ of $\mathbb{Z}$ on A. On the additive part of any ring R, this action coincides with multiplication by image of n in R. Reasoning in terms of scalar multiplication means that we don't have to worry about this identification; often the properties of the action are all we need. Similarly, instead of reasoning about a subring R of a ring S, it is more convenient to view S as an R-algebra. This is equivalent to saying that there is a homomorphism from R to S, namely, the one that maps each element r in R to $r \cdot 1$. Any subring is a homomorphic image of the identity map, so, once again, there is no substantial mathematical difference. But building a library around images of morphisms rather than substructures avoids the need to identify one structure with another; the identification is built into the theorems. (See Baanen et al. [2021] for a nice instance of this.)

In fact, reasoning about structures and mappings between them from a structural algebraic perspective often makes it unnecessary to reason about elements at all. Every subspace of a vector space is the image of a linear map, and it is often natural to build a library around properties of morphisms and maps. There is no need to worry about identifying elements if the theorems are about the identifications themselves.

Finally, there are times when it is impossible to avoid identifying two distinct structures and transferring results between them. Sometimes we really need to identify the real numbers with the one-dimensional vector space over the reals, and sometimes we really need to identify the elements of an algebraic number field with their representations in the complex numbers. In that case, proof assistants often use *casts* or *coercions* from one structure to another. There are various mechanisms that can be used to infer the need for such coercions and insert them automatically, as is done in most programming languages. There are also mechanisms that can be used to translate statements across an embedding or isomorphism [Barthe et al., 2003, Huffman and Kuncar, 2013, Lewis and Madelaine, 2020]. Coercions are generally not required to be injective; so in a sense elements of the source are identified with one another as well as with their images. (See also the discussion of quotients in Subsection 6.3.) Homotopy type theory offers another solution, based on an axiom known as *univalence* [Univalent Foundations Program, 2013].

It may seem disappointing that I have offered a grab-bag of ad-hoc approaches to managing identification, especially when mathematicians seem to do it so effortlessly. Surely, one might think, there must be a simpler explanation as to how mathematicians manage to transfer results from domain to domain without violating foundational norms. But it is a mistake to look for an easy explanation. Mathematics requires extensive training, and learning to do mathematics requires learning how to manage abstraction appropriately. The strategies I have described are all natural, though partial, explanations of what is going on under the hood. The fact that mathematicians have internalized the allowable moves does not mean that we cannot make them explicit. In that respect, formalization efforts can help us understand the inner workings of the practice.

5 Mathematical theories

So far, we have focused on some of the basic components of mathematical statements. These, in turn, are used to formulate definitions, state theorems, and prove them. Collections of definitions and theorems are then structured into mathematical theories. This section considers some of the syntactic mechanisms that the subject relies on to structure knowledge in this way.

5.1 Definitions

Mathematical definitions are often set apart typographically, but they are also often introduced on the fly, in the flow of a mathematical text.

> **Definition 2.** R is said to be a *principle ideal domain (PID)* if every ideal of R is principle.
> [Ireland and Rosen, 1990, p. 9]

> **Definition.** Given a point x of a set X and an open set U of the space Y, let
>
> $$S(x,U) = \{f \mid f \in Y^X \text{ and } f(x) \in U\}.$$
>
> The sets $S(x,U)$ are a subbasis for a topology on Y^X, which is called the *topology of pointwise convergence* (or the *point open topology*).
> [Munkres, 2000, p. 281]

> By a *measure algebra*, we mean a Boolean σ-algebra $\mathcal{A}$ together with a nonnegative real-valued function μ defined on $\mathcal{A}$ such that $\mu(A) = 0$ if and only if $A = 0$ and
>
> $$\mu\left(\bigvee_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu A_i$$
>
> if $A_i \wedge A_j = 0$ for $i \neq j$.
> [Royden, 1988, p. 398]

In either case, the text serves to associate the body of the definition, the *definiens*, to the *definiendum*, which is the word or phrase being defined. In the first and third examples, the body of the definition associates a predicate on the type of objects it describes, in the first case, a ring R, and in the second, a pair $(\mu, \mathcal{A})$ of the kind indicated. In other cases, a definition describes a new object or structure.

Making sense of a definition by description requires recognizing that there is exactly one object meeting the description. When a function is defined in a programming language, the definitional apparatus itself guarantees the existence of a function, or at least a partial function, meeting the specification. Modern mathematics, however, allows for more liberal descriptions, and it is often not obvious that the object described has been well defined. Indeed, it is a common procedure to justify such a definition explicitly by first proving existence and uniqueness. (And sometime uniqueness means uniqueness up to isomorphism, along the lines described in Subsection 4.5.)

Sometimes a definition is a construction—we speak of the construction of Haar measure or the Radon–Nikodym derivative or a direct limit. The definition of a function sometimes comes with an implicit algorithm (see Subsection 8.2), but mathematics tends to favor definitions that characterizes the concept's role in a theory over the means of computation. In other words, definition of the greatest common divisor of two integers or the Legendre symbol may or may not be accompanied by an algorithm to calculate it.

5.2 Theorems

Theorems are the building blocks of theories. They are noteworthy accomplishments in their own right and stepping stones to even greater ones. The word "proposition" is generally reserved for smaller theorems, facts that are generally unsurprising but which form the basis for more substantial results. A "lemma" is a fact established expressly to support the proof of a theorem, and the label "corollary" is used to denote a straightforward consequence.

Setting out the inferential structure of a mathematical theory requires having ways of referring to theorems that have previously been established. In a journal article or textbook, theorems generally have unique identifiers like "Theorem 5.3" or "Proposition 4.2.3," and these can be used across publications, as in "Theorem 4.3 of [5]." Some fundamental results have names, like the prime number theorem, the ergodic theorem, the dominated convergence theorem, and the fundamental theorem of algebra. Some names celebrate their authors, like Birkhoff's theorem, Picard's theorem, Szemerédi's theorem, Whitehead's theorem, Minkowski's inequality, and the Hilbert basis theorem. Some, like the Yoneda lemma, König's lemma, and Zorn's lemma show that apparently minor results can take on added significance after the fact.

Sometimes a theorem is referred to descriptively, like the compactness of S^n, the uniqueness of Haar measure, the existence of an orthogonal complement, the monotonicity of the exponential function, and the fact that a positive semidefinite matrix has real eigenvalues. Using such a description suggests that supporting references are not hard to come by, and that the results are assumed to be part of a common background knowledge.

5.3 Proofs

In mathematical texts, proofs are often introduced with the label "*Proof*" and closed with a box at the end, a notational device introduced by Paul Halmos. The flow of information is generally linear. The primary justification for a claim typically comes from the statement or two preceding it, though additional background knowledge is often needed, and occasionally facts established earlier in a proof. Statements of fact are generally favored over the reasons that they hold, though a number of devices are used to indicate the latter.

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

> [Royden, 1988, p. 151]

> By our previous results the first case obtains when $p \equiv 1 \bmod 4$ and $p \equiv \pm 1 \bmod 12$.
> [Ireland and Rosen, 1990, p. 54]

> Because H is connected, the fiber F over $\gamma(0) = \gamma(1)$ is connected.
> [Duistermaat and Kolk, 2000, p. 175]

> 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.
> [Hatcher, 2002, p. 175]

> 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, $\int f\, d\mu_n = E[f(Y_n)] \to E[f(Y)] \to \int f\, d\mu$.
> [Billingsley, 1995, p. 335]

> Using Hilbert's theorem 90 again, together with the fact that $|\sigma\alpha| = |\alpha|$ for all $\alpha \in K^*$, we see that $H = K^{*1-\sigma}$.
> [Lang, 1994, p. 189]

Sometimes the justification for a claim is an explicitly named theorem, like the bounded convergence theorem or Hilbert's theorem 90. Sometimes it is a named property, like the commutativity of addition or the compactness of a space. Sometimes it involves repeating a fact previously established. Sometimes it requires a few words of explanation, and sometimes it involves reminding the reader of something they already know.

The biggest difference between informal proofs and the proofs that are parsed by proof assistants is the amount of justificatory information that is left implicit in the informal ones. Every fact in a formal library, no matter how mundane or trivial, has a unique identifier, and one of the hardest parts about using a proof assistant is that one generally has to learn the names of facts in the library and refer to them explicitly when needed. Justificatory steps are also spelled out in far greater detail in a proof assistant, whereas ordinary mathematics has fewer resources for naming facts and composing them. Informal proofs rely instead on contextual hints and cues that enable a reader with sufficient expertise to spell out the details. This helps explain why expertise is so important; checking an informal proof is far from a mechanical process.

Sometimes the justification for a later inference is flagged in advance.

> Note that adjoining a copy of $\mathbb{Z}$ induces an injection on π_1 since the induced homomorphism is the free product of the injection $\pi_1(A) \longrightarrow \pi_1(\mathbb{Z})$ with the identity map on the complementary free factor.
> [Hatcher, 2002, p. 172]

> Note that each set U_n' is open, being the difference of an open set U_n and a closed set $\bigcup_{i=1}^{n} \overline{V}_i$.
> [Munkres, 2000, p. 201]

Consider the significance of *noting* as a cognitive act; the proof asks the reader to keep the fact close at hand so that it can be deployed when needed, often without further comment. Mathematical texts call on us to perform other cognitive acts as well:

> Expressing each generator as a sum of homogeneous elements, . . .
> [Lang, 2002, p. 428]

> We can choose the indices i in such a way that $\sigma_i A_1 = A_i$.
> [Lang, 2002, p. 183]

> Now identify $\mathfrak{g}$ with $\mathbb{R}^3$ by means of an orthonormal basis with respect to the Ad G-invariant inner product on $\mathfrak{g}$.
> [Duistermaat and Kolk, 2000, p. 169]

> . . . consider the n simplex $[v_0, \cdots, v_n]$ with v_0 at the origin and v_k the unit vector along the kth coordinate axis for $k > 0$.
> [Hatcher, 2002, p. 270]

> We now apply the bilinear form of Parseval's identity (Lemma 1.5, Chapter 3) to the integral defining $\mathcal{A}$.
> [Stein and Shakarchi, 2003, p. 104]

To make our way through a proof, we have to express things, choose things, identify things, consider things, and apply things. Traditional logic doesn't account for these acts, but only the results of performing them, which should be a network of inferences that takes us from a theorem's premises to its conclusion.

Logicians are fond of representing proofs diagrammatically as trees that branch at inference rules that have more than one premise. But the flow through an ordinary mathematical proof is remarkably linear. Sometimes the application of a theorem requires checking side hypotheses that are dispensed with quickly, and the base case in a proof by induction is usually dismissed easily. Occasionally one comes across a case split where the cases have equal footing, and the in those situations, the branching nature of the proof is made clear.

> If $d \equiv 2$ or 3 (4) then. . .
> If $d \equiv 1(4)$ . . .
> [Ireland and Rosen, 1990, p. 191]

> There are three cases. If $m < n$ then $f^{-1}(C)$ closed implies $C \cap X^{n-1}$ closed, hence $C \cap \bar{e}^m_\beta$ is closed since $\bar{e}^m_\beta \subset X^{n-1}$. If $m = n$ then e^m_β is one of the cells e^n_α, so $f^{-1}(C)$ closed implies $f^{-1}(C) \cap D_\alpha$ is closed, hence compact, hence its image $C \cap \bar{e}^n_\alpha$ under f is compact and therefore closed. Finally there is the case $m > n$.
> [Hatcher, 2002, p. 521]

Such case splits, however, are few and far between.

Although a proof is generally presented as a sequence of factual assertions, the narrative has to pull off the difficult task of enabling us to determine how each successive statement follows from what has come before. Mathematical language supplies an array of devices that can be used to suggest inferential patterns, flag relevance, and indicate the necessary connections. Some of these devices are implemented in proof assistants. The Mizar proof language [Grabowski et al., 2010] was a pioneering effort in this sense, providing the first formal representation of informal proof vernacular. Most proof languages today contain similar means for structuring formal proofs. Those who design the languages have to balance competing goals, such as making proofs easy for users to write, making proofs easy for users to read, and making proofs easy for the system to process. But there is still more to be learned, and paying closer attention to the literature should provide better insight as to how well-chosen language supports mathematical inference.

5.4 Notation

Mathematical texts often introduce new notation.

> Let V, V' be two vector spaces, and suppose given a mapping
>
> $$V \times V' \to K$$
>
> denoted by
>
> $$(x, x') \mapsto \langle x, x' \rangle$$
>
> for $x \in V$ and $x' \in V'$. We call the mapping *bilinear* if . . .
> [Lang, 2002, p. 144]

> We define the *convolution* $f \star g$ on $[-\pi, \pi]$ by . . .
> [Stein and Shakarchi, 2003, p. 44]

> We define the *Dirichlet L-function associated to* χ by the formula
>
> $$L(s, \chi) = \sum_{n=1}^{\infty} \chi(n) n^{-s}.$$
>
> [Ireland and Rosen, 1990, p. 254]

Exploring what makes for good notation would take us too far afield, but it is worth recognizing the good notation has remarkable saying power [Cajori, 1993]. Even fonts and letter choices tend to stick around: the use of π goes back to the early eighteenth century, and the use of $f(x)$ and $g(x)$ for functions in analysis, as well as the use of e, i, and φ for the familiar mathematical constants, goes back to Euler. We still speak of Riemann's ζ function and a Dirichlet L series, and we tend to use Fraktur letters $\mathfrak{a}$, $\mathfrak{b}$, and $\mathfrak{c}$ for ideals, following Dedekind.

One challenge for proof assistants is that notation, like identifiers, can be overloaded. The addition symbol in $x + y$ can refer to addition in the integers or reals, but also to addition in any ring or additive group. The addition symbol can also be used to denote the pointwise sum $f + g$ of two functions, or the sum $\mu + \nu$ of two measures.

When a symbol is used in two unrelated ways, computer scientists refer to it as *ad hoc* polymorphism. In all the examples above, one can view the domains over which the addition is defined as being instances of an additively written semigroup, which makes them instances of *parametric polymorphism*. We can therefore interpret them as the same addition, where only the semigroup itself is left implicit. We will return to this when we discuss algebraic language in Section 7. Computational implementations of mathematical language use various mechanisms to infer the relevant structure. Even ad-hoc uses of an addition symbol can be used as instances of an "addition structure" [Wadler and Blott, 1989].

5.5 Calculation

Once seen as the essence of mathematics, calculation is now often associated with applied mathematics, but it plays a role in pure mathematics as well. Here is an excerpt from a proof of the ergodic theorem:

$$\begin{aligned}
\int_{[|a_n|>\lambda]} |a_n|\, dP &\leq \frac{1}{n}\sum_{k=1}^{n}\int_{G_\lambda} |f(T^{k-1}\omega)|P(d\omega) \\
&\leq \frac{1}{n}\sum_{k=1}^{n}\left(\int_{|f(T^{k-1}\omega)|>\alpha} |f(T^{k-1}\omega)|P(d\omega) + \alpha P(G_\lambda)\right) \\
&= \int_{|f(\omega)|>\alpha} P(d\omega) + \alpha P(G_\lambda) \\
&\leq \int_{|f(\omega)|>\alpha} P(d\omega) + 2\frac{\alpha}{\lambda}E[|f|].
\end{aligned}$$

[Billingsley, 1995, p. 318]

In this case the calculation is a chain of equalities and inequalities. Formal systems rarely specify syntax for calculations arguments or distinguish them from other types of arguments, but proof assistants generally provide special mechanisms to support them.

6 Set-theoretic abstraction

Sets, functions, and relations are so fundamental to contemporary mathematics that it is easy to forget that they are recent additions to the subject. In the middle of the eighteenth century, Euler grounded calculus on the notion of a function, but the term did not take on anything approximating its contemporary significance until late in the nineteenth century. Similarly, it is hard to find anything like the modern notion of a set before that time.

The goal of this section is to characterize the contemporary notions of set and function and consider their role in mathematical language. In large part, set-theoretic language evolved in the nineteenth century as a means of supporting algebraic abstraction, which is discussed in the next section.

6.1 Functions

In the early nineteenth century mathematicians dealt with a number of mathematical objects that we view as instances of the general function concept today, but were not then seen as such. These include sequences, series, geometric transformations, and permutations of roots of an equation. The word *function* and its cognates in western languages were used exclusively for functions that are defined on the real or complex numbers and take values in one of those domains [Monna, 1972, Youschkevitch, 1976/77, Avigad and Morris, 2014]. Even number-theoretic functions, like the Euler ϕ function or the totient function, were described as *symbols* or *characters* rather than functions. The general view of a function as a mapping between any two domains first emerged later in the nineteenth century, in Dedekind's supplements to Dirichlet's *Lectures on Number Theory* and Frege's *Begriffsschrift*.

The modern concept of function has these features:

1. A function $f : A \to B$ maps elements from a domain A to a codomain B.
2. Any description that associates a unique element of B to each element of A serves to define a function. There need not be any algorithm or procedure for producing an element of B from an element of A; in fact, there need not be any means of representing particular elements of A in a way that an algorithm can act on them.
3. The axiom of choice allows us to consider functions for which we don't even have a means of description.
4. Two functions from A to B are considered to be the same when they take the same values at every argument. There is nothing more to a function than the association of output values to inputs.

Foundational accounts differ as to whether the specification of the codomain is part of the function, for example, whether the function $f(x) = x^2$ from $\mathbb{R}$ to $\mathbb{R}$ is simultaneously a function from $\mathbb{R}$ to the nonnegative reals. Mathematicians freely identify the two. Logical foundations often differ as to whether applying a function f to an element outside its intended domain is a syntactic error or a semantic one, that is, whether it is grammatically incorrect or only mathematically dubious.

In set theory, a function $f : A \to B$ is usually defined to be a set of ordered pairs, where $f(x) = y$ means that the pair (x, y) is an element of f. A function $f(x, y, z)$ with multiple arguments is generally viewed as a function $f : A \times B \times C \to D$. In type-theoretic foundations, the notion of a function $f : A \to B$ is a primitive notion, and the type of inputs and the type of the outputs are both an inherent part of the syntactic specification. We have seen that in type theory, a function $f(x, y, z)$ is commonly represented as an object of type $A \to (B \to (C \to D))$ and application is commonly expressed by writing $f\,x\,y\,z$, where x, y, and z are of type A, B, and C, respectively. This has the sometimes useful side effect that an expression like $f\,x\,y$ denotes an object of type $C \to D$, in which case, f is said to be *partially applied*. In type theory, it is harder to view a function that takes values in the integers as simultaneously a function that takes values in the real numbers. In compensation, the fact that the type of an expression like $f\,x$ is unambiguous is often useful.

The modern viewpoint allows us to treat functions as objects in their own right, on par with other mathematical objects, like natural numbers. We can quantify over functions the same way we quantify over numbers, we can define structures whose elements are functions just as we can define structures whose arguments are numbers, and we can define functions (also known as *functionals*) whose arguments are functions. This is what distinguishes the discussion of functions here from the discussion of operations in Subsection 3.2. We can talk about multiplication as an operation on the integers, but when we talk about the multiplication *function* on the natural numbers, we are talking about a first-class object. Any implementation of mathematical language has to mediate between these two views, that is, the treatment of functions as objects and the treatment of functions as operations in the sense of Subsection 3.2.

6.2 Sets

In his book, *Labyrinths of Thought*, José Ferreiros observes that in the nineteenth century there was no standard terminology to denote the set concept [Ferreirós, 1999, page xx]. Although

French quickly settled on *ensemble*, Italian on *insieme*, and Spanish on *conjunto*, German terminology was not as uniform. Dedekind used words like *System* and *Gebiet*, while Cantor tended to favor *Mannigfaltigkeit* and *Menge*. It is interesting to read the following snippet from a letter from Dedekind to Cantor.

> (incidentally, I should like to see the shorter and equally Riemannian word "domain" [GebietJ given clear preference over the clumsy word "manifold" [Mannigfaltigkeit]. . .)
>
> [Ewald, 1996, p. 870]

The term *manifold* has, of course, survived, though it is used to describe sets with a specific geometric structure. Language and notation for sets in general is found everywhere in mathematics.

> Let $A_0 \subset C_0$ be the set of cubes disjoint from K, and inductively, let $A_k \subset C_k$ be the set of cubes disjoint from K and not contained in cubes of A_j for $j < k$.
>
> [Hatcher, 2002, pp. 525–526]

> Consider the set of ideals $\{A^i \mid 1 \leq i \leq h_F + 1\}$.
>
> [Ireland and Rosen, 1990, p. 179]

> W acts transitively on the set of connected components of $G^{\mathrm{reg}} \cap T$.
>
> [Duistermaat and Kolk, 2000, p. 154]

> Let $\{e_1, \ldots, e_m, \ldots\}$ be a basis of F. . .
>
> [Lang, 2002, p. 521]

> A set A is a μ-continuity set if it is a Borel set and $\mu(\partial A) = 0$.
>
> [Billingsley, 1995, p. 335]

Talk of sets is often implicit in the definitions of mathematical spaces and structures because they are often defined by specifying the underlying set.

> The space of bounded linear functionals on a normed space X is called the *dual* (or conjugate) of X and is denoted X^*.
>
> [Royden, 1988, p. 326]

> Given a map $f : A \to B$, let E_f be the space of pairs (a, γ) where $a \in A$ and $\gamma : I \to B$ is a path in B.
>
> [Hatcher, 2002, p. 407]

> Let $[M]$ denote the isomorphism class of a finite module M. We define the sum to be the direct sum. Then the isomorphism classes over the ring form a monoid.
>
> [Lang, 2002, p. 139]

Any implementation of mathematical language has to make it easy to describe sets and reason about them.

In axiomatic set theory, the notion of set is basic; everything is a set, and a function is a set of ordered pairs. In axiomatic type theory, in contrast, the notion of a function between types is basic. A predicate on a data type α is viewed as a function from α to truth values, and sets are in one-to-one correspondence with predicates: given a predicate like *Even* on the natural numbers, we can form the set of even numbers, and, of course, given any set s, we can consider the property of being an element of s.

There in an important difference between set theory and type theory. In set theory, any collection of objects can form a set. We can consider the set consisting of the number π, the multiplication function on the integers, and the Eiffel tower. In type theory, any set is a set of objects of some fundamental type. We can consider a set of integers or a set of functions, but there is no notion of a set of *arbitrary* mathematical objects. Whereas this is mathematically natural, the distinction between sets and types in type theory forces us to distinguish between the integers as a fundamental type and the integers as a subset of the real numbers. But, as we have seen in Section 4, the problem of identifying objects and structures is not limited to type theory. It poses challenges for set-theoretic foundations as well.

6.3 Quotients

In a familiar set-theoretic construction, if A is a set and $\equiv$ is an equivalence relation on A, the quotient $A/\equiv$ is defined to be the set of equivalence classes of A. Functions and relations on A that respect the equivalence relation descend to functions and relations on $A/\equiv$. (Confusingly, computer scientists often use the term "lift" instead of "descend.") This construction provides a useful way to treat equivalent objects as identical.

Quotient constructions come up often.

> We denote by $P_\mathfrak{c}$ the subgroup of P consisting of those principle fractional ideals (α) with $\alpha \in k_\mathfrak{c}$. Then it is clear that $P_\mathfrak{c}$ is a subgroup of $I(\mathfrak{c})$. The factor group $I(\mathfrak{c})/P_\mathfrak{c}$ will be called the group of $\mathfrak{c}$-*ideal classes*.
> [Lang, 1994, p. 125]

> From the diagram above, $H_n(X)$ can be identified with $H_n(X^n)/\mathrm{Im}\ \delta_{n+1}$.
> [Hatcher, 2002, p. 140]

> Let X be the mapping torus of f, the quotient space of $(S^2_\alpha \vee S^2_\beta) \times I$ under the identifications $(x, 0) \sim (f(x), 1)$.
> [Hatcher, 2002, p. 589]

> Let $\mathfrak{M}$ be the class of measurable subsets of $[0, 1]$, $\mathfrak{N}$ the class of subsets of measure zero, and m Lebesgue measure. Then $\langle \mathfrak{M}/\mathfrak{N}, m\rangle$ is a separable measure algebra without atoms.
> [Royden, 1988, p. 399]

> The equivalence classes of fractional ideals form a finite group, . . . which we call the *ideal class group*.
> [Lang, 1994, p. 123]

> Because the canonical projection $\pi : M \to G \setminus M$ is continuous and maps open subsets of M onto open subsets of $G \setminus M$, the orbit space $G \setminus M$ is locally compact.
> [Duistermaat and Kolk, 2000, p. 104]

Mathematicians often pass silently between talking about elements of a quotient and talking about representatives in the quotiented set. For example, it is common to conflate functions with equivalence classes of functions up to almost everywhere equivalence in a function space.

7 Algebraic abstraction

Algebra involves calculating with variables that range over elements of some domain, using rules that are valid for that domain. The practice was used in the late sixteenth century by Viète, who saw the algebraic method as the foundation for a universal science. In the seventeenth century, Descartes and Leibniz held similar views.

We obtain modern algebra by making the rules explicit and abstracting the underlying domain, so that variables range over any domain that obeys the general laws and the theorems we prove can be applied to any such domain. In other words, we get modern algebra (also known as *abstract algebra*) when we treat the objects under consideration as elements of an algebraic structure. This is a powerful means of abstraction that makes the properties that are relevant to a particular reasoning context explicit.

Conventional terminology wavers between using the expression "algebraic structure" to describe a class of mathematical structures and using it to describe any particular instance. In other words, sometimes we call the group concept an algebraic structure and sometimes we call a particular group an algebraic structure. For clarity, here I will reserve the term "structure" for the individual instances, and use the phrase "abstract structure" for the axiomatically characterized concept. With that terminology in place, we can describe the general practice: we define an abstract structure by specifying a linguistic signature and a collection of properties, and this allows us to reason uniformly about all its instances. The process of reasoning about such algebraically characterized classes becomes an art unto itself, and it is worthwhile to consider how mathematical language supports it.

7.1 Implicit structure

An abstract structure is specified with a signature. We start by specifying that it has one or more carrier sets, and then we specify the functions and relations that are to be found in any instance, together with the axioms they are assumed to satisfy. A group $(G, \circ, e, \cdot^{-1})$ is any structure with a carrier set, G, an element $e \in G$, a binary operation $\circ$ on G, and a unary operation $\cdot^{-1}$ on G, all satisfying the group axioms. A partial order (X, R) consists of a transitive, reflexive, and antisymmetric relation R on a set X. An abstract structure can sometimes refer to higher-order objects over the carrier sets. A topological space consists of a pair $(X, \mathcal{T})$ where $\mathcal{T}$ is a collection of subsets of X satisfying the axioms for a topological space, and a hypergraph (X, E) consists of a set X and an arbitrary collection E of nonempty subsets of X.

When a mathematical object is an element of the carrier set of a structure, properties ascribed to the object often implicitly invoke the signature of the ambient structure. An element $a \in X$ of a partial order (X, R) is *minimal* if it is R-related to every other element of X. Similarly, properties of a function $f : X \to Y$ between carrier sets of two structures or a subset $A \subseteq X$ of the carrier set of a structure often make implicit reference to those structures. Saying that a set $A \subseteq X$ is *closed* or *compact* presupposes that there is a topology on X, saying that $f : X \to Y$ is continuous presupposes that there are topological structures on X and Y. This is an important sense in which sorts make a difference: to make sense of a statement like "a is minimal" or "f is continuous," it is not enough to view a as an object and f as a function; we have to view them as objects of appropriate sorts, and we have to view those sorts as coming equipped with ambient structures.

In any mathematical text, the relevant structures are determined by context. The same can be said for operations like the maximum function on an order or the closure operation on sets in a topology, and for notation like $x \cdot y$ and x^y. In proof assistants, various methods are used to infer implicit structure [Wenzel, 1997, Asperti et al., 2009, Sozeau and Oury, 2008, Garillot et al., 2009, Selsam et al., 2020]. A good semiformal description should clarify the means by which the relevant associations are registered by a mathematical text.

7.2 Algebraic hierarchies

Abstract structures bear relationships to one another. An abstract structure can add axioms to one described previously. A *total order* is a partial order (X, R) such that for any two elements a and b of X, either $R(a, b)$ or $R(b, a)$ holds. A *well-founded partial order* adds a different assumption, and a *well-founded total order* retains both. An *abelian group* is a group in which the group operation is commutative. Adding an axiom to an abstract structure means that the resulting class of instances is contained in the class of instances of the prior one, which is to say, any instance of class of structure with the additional axiom is an instance of the class without it. This is one way in which structures form a hierarchy.

An abstract structure can also extend another by adding additional data. For example, a *pointed topological space* is just a topological space $(X, \mathcal{T})$ with an additional point $a \in X$. More commonly, an abstract structure will extend another with both additional data and additional axioms. A *group* is a monoid structure with an inverse; an *ordered group* is a group is together with a partial order on the carrier that is compatible with the group operation. A *ring* $(R, \cdot, +, 1, 0)$ is best described as the combination of a monoid $(R, \cdot, 1)$ and an abelian group $(R, +, 0)$ satisfying the distributivity axioms.

Viewing a network of abstract algebraic structures and the relationships between them as a hierarchy supports modularity and abstraction in the sense of Section 4. A definition made in the context of one abstract structure instantly carries over to any others that inherit that structure, and a theorem proved about an abstract structure immediately applies to any of its instances. The reason that this is relevant to our discussion of mathematical language is that the transfer is built in to the grammar. Any concepts defined in the context of a group can be interpreted in any abelian group or any ordered group, and, in practice, such concepts are deployed without comment or explanation. So a semiformal description of mathematical language has to specify not only how implicit algebraic structure is inferred, but also how algebraic relationships are tracked and used to make sense of mathematical statements.

The combined need to infer structure and to track relationships in the algebraic hierarchy gives rise to an interesting problem, namely, the *diamond problem*, that has been made manifest by the practice of formalization. It arises from the fact that, given a complex hierarchy, structure can be inferred in multiple ways, and the coherence of the practice relies on the fact that the results agree.

Here is an example of the phenomenon. Every metric space is an instance of a topological space; given a metric space, there is a unique topological space that is determined by the metric space. Given topological spaces with carriers X and Y, there is a canonical topology on their product, $X \times Y$. If X and Y are metric spaces, there are various natural ways of imposing a metric on $X \times Y$ that induces the same topology. Suppose we fix one, let f be a function from $X \times Y$ to some other space, and say that f is continuous. We have seen that this requires us to view $X \times Y$ as a topological space. The question is: what is the topological space that is implicit in the statement?

The problem is that there are two ways to find such a space. Given that X and Y are metric spaces, we can view them as topological spaces, and then take the canonical topology on $X \times Y$. Alternatively, we can view $X \times Y$ as a metric space, and take the metric-space topology on $X \times Y$. Now, assuming we have chosen the metric on $X \times Y$ to have this property, the two choices coincide. But it may be a nontrivial mathematical *theorem* that they do. Suppose, in proving a theorem, we establish the continuity of f using one understanding of the relevant topological space, and then apply a theorem that presupposes the continuity of f using the other. We can hardly expect our proof assistant to run off and prove a substantial mathematical theorem just to infer that the two statements are the same. This is yet another instance of the identification problem raised in Section 7: mathematicians are comfortable shifting between equivalent readings of a statement once they have internalized the equivalence, but explaining what is going on formally requires some effort.

One solution found in the literature [Buzzard et al., 2020, Affeldt et al., 2020] is to take the structure of a metric space to include a specification of the associated topology, together with a proof that the topology is compatible with the metric. That makes it possible to ensure that both paths yield answers that are easily seen to be the same. But more experimentation is needed to determine whether this is a viable way of solving all the diamond problems that crop up in practice, and whether the method can be made principled and general enough to become part of our semiformal specification.

7.3 Structures as objects

What gives an entity the status of a mathematical *object*? Section 3 presented some of the hallmarks: variables can range over objects, objects bear properties and relationships to one another, and mathematical operations build new objects from old ones. Section 4 provided more: objects can be identified with other objects in various ways. The current section provides yet another aspect of objecthood: mathematical objects are, or can be seen as, elements of axiomatically characterized structures.

In contemporary mathematics, algebraic structures themselves are mathematical objects in all these senses. We quantify over groups, rings, and fields the same way we quantify over numbers and points on the Euclidean plane. We talk about products, powers, and limits of sequences of algebraic structures the same way talk about products, powers, and limits of sequences of numbers. Various operations can be used to build new structures out of old ones; the automorphisms of a group, ring, or field form group, the square-integrable real-valued functions on any measure space form a Hilbert space, and for every field F and natural number n, the structure F^n is a vector space. We identify structures, though here the relevant notion of "sameness" is isomorphism, which is coarser than the notion of equality that derives from viewing a structure as a tuple consisting of a carrier set and some functions and relations. And algebraic structures can be inhabitants of larger algebraic structures. The category of abelian groups (or abelian groups in a fixed set-theoretic universe) is an algebraic structure whose elements are abelian groups, a sheaf of rings is a collection of rings bearing an algebraic structure, and measure-theoretic probability allows us to consider measures on spaces whose elements are themselves measures.

This means that algebraic structures play dual roles. We can treat a group G as a mathematical object, and at the same time consider elements x and y of G. In the latter case, G specifies the sorts of objects that x and y are, and this is used to make sense of expressions that involve them. This duality is handled in various ways in proof assistants. One approach is to take a group, G, to consist of a structure, one component of which is the carrier type, $\mathsf{carrier}\ G$. The statement that x and y are elements of G is interpreted as shorthand for saying that x and y are elements of $\mathsf{carrier}\ G$ by coercing G to its carrier. An alternative is to use *type classes*, which provide a means of using context to associate information with a given data type. In that case, the statement that G is a group is interpreted as saying that G is a data type with an associated group structure. The statement that x and y are elements of G is interpreted as just that; but an expression like $x \cdot y$ is interpreted as making reference to an ambient structure that is inferred from the context.

8 Other aspects of mathematical language

There are a number of important aspects of mathematical language that we have not even begun to address. We use language to state conjectures, raise questions, launch research programs, describe methods, and assess the value of mathematical contributions and artifacts. In this final section, we briefly consider a few additional aspects of mathematical communication.

8.1 Diagrams

Diagrams and images are important components of mathematical language. They are used to justify and explain, often more efficiently and effectively than text-based representations. In the last couple of decades, there has been a substantial literature on uses of diagrams in reasoning; see Giaquinto [2007], Manders [2008], De Toffoli and Giardino [2014], De Toffoli [2017], Hamami and Mumma [2013] for a representative sample. Work like this provides a diagrammatic analogue of the type of textual analysis proposed here, by developing a taxonomy of representational devices and their uses.

Given the size of the literature on mathematical diagrams, it is reasonable to ask why the analysis of textual communicative devices seems to be lagging. One possibility is that the differences between diagrammatic representations and formal logic are so dramatic that it is clear that there is something interesting going on, whereas, in contrast, it is easier to convince ourselves that formal logic captures the essence of text-based reasoning. I hope this chapter has made the case that the mechanisms employed in text-based reasoning are just as elaborate, subtle, and mysterious as the ones employed in diagrammatic reasoning, and that they are just as worthy of study, if not more so.

8.2 Algorithms

For most of its history, mathematics was algorithmic in nature. Euclid's *Elements* provided recipes for constructing geometric objects, early algebraists were concerned with recipes for calculating quantities of interest, early probabilists were concerned with calculating odds in games of chance, and early analysts were concerned with calculating the motions of the planets.

Pure mathematics tends to keep computation at arm's length, relegating questions about how to calculate quantities described or guaranteed to exist by a theory to the realm of application. A contemporary textbook on modern algebra might well prove that every matrix can be put in Jordan canonical form without saying explicitly how to do it [Lang, 2002, p. 559]. Still, algorithms sometimes creep into textbooks in pure mathematics.

As a first application of quadratic reciprocity we show how, in conjunction with Proposition 5.1.2, it can be used in numerical computations of the Legendre symbol.
[Ireland and Rosen, 1990, p. 54]

Example 4L.3: Stable Splittings. The formula ($\star$) tells us how to compute Steenrod squares for $\mathbb{R}P^\infty$, hence also for any suspension of $\mathbb{R}P^\infty$.
[Hatcher, 2002, p. 491]

Numerical and symbolic computation are an important part of mathematics, and the language that textbooks and journal articles use to describe such algorithms should be viewed as an important part of mathematical language.

Fortunately, computer science has developed ample means of representing algorithms formally—this is precisely what programming languages do. And proof assistants are commonly used for software verification, so there is a vast literature to draw on when thinking about mathematical algorithms.

8.3 Plans

Our discussion of proof in Subsection 5.3 ignored a number of narrative elements that are found in mathematical texts. A proof serves as a tour guide, drawing our attention along, highlighting key facts, and shaping the experience of understanding. A good proof reminds us where we have been, and it gives us a sense of where we are going and what we need to have with us on the way.

Now let $\mathfrak{c}$ be a connected component of $\mathfrak{t}^{\mathrm{reg}}$.
[Duistermaat and Kolk, 2000, p. 147]

Think about the significance of the word “now.” It is hard to pin down its inferential significance, but it plays an important role. It tells the reader that the part that has come before has been preparatory to this moment; it asks us to check our footing, and prepare ourselves for the next phase of the ascent.

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.
[Ireland and Rosen, 1990, p. 181]

Thus it only remains to show that (ii) $\Rightarrow$ (iii). To do this. . .
[Royden, 1988, p. 206]

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.
[Ireland and Rosen, 1990, p. 251]

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.
[Lang, 1994, p. 63]

Rebecca Morris has observed that we expect a proof to be *motivated*, in the sense that the structure of such a proof should make it clear how each step might have been anticipated, and how each step gets us closer to the goal [Morris, 2020]. Her work explores some of the devices that contribute to that aim. Morris and Yacin Hamami have argued that good proofs show evidence of a rational *plan* [Hamami and Morris, to appear]. We tend to understand other people's behavior by assuming that they have particular goals and by interpreting their actions in relation to plans that achieve those goals. In a similar way, we make sense of proofs by interpreting them in terms of a systematic overall plan. Proof assistants give us languages for expressing definitions, theorems, and proofs. What can we say about the components of mathematical language that are used to put forth a plan?

8.4 Intuitions and heuristics

Seventeenth century mathematicians like Descartes and Leibniz were enamored of the prospect of having methods of solving mathematical problems, like geometric construction problems and optimization problems. One can view this as a search for computational algorithms. But one can view their interest more broadly as having a bodies of *heuristics* that limit the search space and guide reason to its desired end. The mathematical literature is filled with such offerings.

> The methods to be used are identical with those already developed in the previous sections.
> [Ireland and Rosen, 1990, p. 103]

> Two corollaries, interesting in themselves, will make clearer the structure of the proof of sufficiency given above.
> [Billingsley, 1995, p. 350]

> The next section treats a general scheme for dealing with path-function questions by in effect replacing an uncountable time set by a countable one.
> [Billingsley, 1995, p. 503]

> We now return, for the remainder of the chapter, to the consideration of various methods for imposing topologies on sets.
> [Munkres, 2000, p. 112]

> The notions of a Galois extension and a Galois group are defined completely algebraically. Hence they behave formally under isomorphisms the way one expects from objects in any category.
> [Lang, 2002, p. 264]

> This section contains a general discussion of invariant densities, especially on homogenous spaces. Although in the next section this is only used in a relatively simple case, the general discussion will be useful for future reference.
> [Duistermaat and Kolk, 2000, p. 179]

> Notice that if f already sends some vertices of K to vertices of L then we may choose g to equal f on these vertices, and hence the homotopy from f to g will be stationary on these vertices. This is convenient if one is in a situation where one wants maps and homotopies to preserve basepoints.

[Hatcher, 2002, p. 179]

Statements like this do not make precise mathematical claims. Instead, they convey information that is meant to shape the way we organize, categorize, and store the knowledge we have, and the way we deploy that knowledge in the future.

Automated reasoning depends crucially on using heuristics to limit the space of options and fruitfully guide a search through a combinatorial explosion of possibilities. Successful machine learning algorithms seem to acquire representations that also structure spaces of options in fruitful ways. If we focus on mathematical statements like the ones above and think about what they are doing, we may be in a better position to design systems that provide better support for mathematical reasoning.

9 Conclusions

Understanding mathematical language can help us develop better means of formalizing mathematics, design better automated reasoning tools, and develop ways of harvesting data from mathematical texts. But it is also interesting in its own right, and it contributes to a better appreciation of mathematics itself. A good piece of mathematics is a work of art, and there is a deep satisfaction to be had in admiring the creativity and insight that went into it. But a good piece of mathematics is also *well designed*, and uncovering the principles that make for good design can help us appreciate the skill and expertise that has gone into it. This chapter makes only a small start at mapping out the landscape, and there is a lot that needs to be done.

Acknowledgment. I am grateful to Gihan Marasingha for helpful suggestions.

References

R. Affeldt, C. Cohen, M. Kerjean, A. Mahboubi, D. Rouhling, and K. Sakaguchi. Competing inheritance paths in dependent type theory: A case study in functional analysis. In N. Peltier and V. Sofronie-Stokkermans, editors, *International Joint Conference on Automatated Reasoning (IJCAR) 2020*, pages 3–20. Springer-Verlag, Berlin, 2020. 27

A. Asperti, W. Ricciotti, C. S. Coen, and E. Tassi. Hints in unification. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 84–98. Springer-Verlag, Berlin, 2009. 4, 26

J. Avigad. The mechanization of mathematics. *Notices of the American Mathematical Society*, 65(6):681–690, 2018. 4

J. Avigad. Modularity in mathematics. *The Review of Symbolic Logic*, 13(1):47–79, 2020. 9

J. Avigad and J. Harrison. Formally verified mathematics. *Communincations of the Association for Computing Machinery*, 57(4):66–75, 2014. 4

J. Avigad and R. Morris. The concept of "character" in Dirichlet's theorem on primes in an arithmetic progression. *Archive for History of Exact Sciences*, 68(3):265–326, 2014. 21

A. Baanen, S. R. Dahmen, A. Narayanan, and F. A. E. Nuccio Mortarino Majno di Capriglio. A formalization of Dedekind domains and class groups of global fields. In L. Cohen and C. Kaliszyk, editors, *Interactive Theorem Proving (ITP) 2021*, volume 193 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 15

G. Barthe, V. Capretta, and O. Pons. Setoids in type theory. *Journal of Functional Programming*, 13(2):261–293, 2003. 15

P. Benacerraf. What numbers could not be. *Philosophical Review*, 74(1):47–73, 1965. 10

Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development – Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag, Berlin, 2004. 1

P. Billingsley. *Probability and Measure*. John Wiley & Sons, Inc., New York, third edition, 1995. 5, 18, 21, 23, 30

J. Blanchette and A. Mahboubi, editors. *Handbook of Proof Assistants and their Applications in Mathematics and Computer Science*. Springer-Verlag, Berlin, to appear. 4

K. Buzzard, J. Commelin, and P. Massot. Formalising perfectoid spaces. In J. Blanchette and C. Hritcu, editors, *Certified Programs and Proofs (CPP) 2020*, pages 299–312. ACM, 2020. 27

F. Cajori. *A History of Mathematical Notations*. Dover Publications, Inc., New York, 1993. Two volumes. Reprint of the 1928 and 1929 originals. 20

M. Cramer, B. Fisseni, P. Koepke, D. Kühlwein, B. Schröder, and J. Veldman. The Naproche project: Controlled natural language proof checking of mathematical texts. In N. E. Fuchs, editor, *Workshop on Controlled Natural Language (CNL) 2009.*, pages 170–186. Springer-Verlag, Berlin, 2009. 2

L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean theorem prover (system description). In A. P. Felty and A. Middeldorp, editors, *Conference on Automated Deduction (CADE) 2015*, pages 378–388. Springer-Verlag, Berlin, 2015. 1

S. De Toffoli. 'Chasing' the diagram—the use of visualizations in algebraic reasoning. *The Review of Symbolic Logic*, 10(1):158–186, 2017. 28

S. De Toffoli and V. Giardino. Forms and roles of diagrams in knot theory. *Erkenntnis*, 79(4): 829–842, 2014. 28

R. Dedekind. Letter to Heinrich Weber (24 January 1988). Translated in Ewald [1996], Volume 2, pages 834–835, 1888. 11

J. J. Duistermaat and J. A. C. Kolk. *Lie Groups*. Springer-Verlag, Berlin, 2000. 18, 19, 23, 24, 29, 30

Euclid. *Euclid's Elements*. Green Lion Press, Santa Fe, NM, 2002. The Thomas L. Heath translation, Edited by Dana Densmore. 4, 9

W. Ewald. *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*. Clarendon Press, Oxford, 1996. in two volumes. 23, 32

J. Ferreirós. *Labyrinth of Thought: A History of Set Theory and its Role in Modern Mathematics*. Birkhäuser Verlag, Basel, 1999. 22

M. Ganesalingam. *The Language of Mathematics: A Linguistic and Philosophical Investigation*. Springer-Verlag, Heidelberg, 2013. 2

F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau. Packaging mathematical structures. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 327–342. Springer-Verlag, Berlin, 2009. 4, 26

T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar, and M. Norrish. Tactictoe: Learning to prove with tactics. *Journal of Automated Reasononing*, 65(2):257–286, 2021. 4

M. Giaquinto. *Visual Thinking in Mathematics: An Epistemological Study*. Oxford University Press, Oxford, 2007. 28

G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. L. Roux, A. Mahboubi, R. O'Connor, S. O. Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, and L. Théry. A machine-checked proof of the odd order theorem. In S. Blazy, C. Paulin-Mohring, and D. Pichardie, editors, *Interactive Theorem Proving (ITP) 2013*, pages 163–179. Springer-Verlag, Berlin, 2013. 15

A. Grabowski, A. Kornilowicz, and A. Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010. 1, 19

T. C. Hales. Formal proof. *Notices of the American Mathematical Society*, 55(11):1370–1380, 2008. 4

Y. Hamami and R. Morris. Plans and planning in mathematical proofs. *Review of Symbolic Logic*, to appear. 30

Y. Hamami and J. Mumma. *Prolegomena* to a cognitive investigation of Euclidean diagrammatic reasoning. *Journal of Logic, Language and Information*, 22(4):421–448, 2013. 28

J. Harrison. HOL light: an overview. In *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 60–66. Springer-Verlag, Berlin, 2009. 1

A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002. 18, 19, 23, 24, 29, 31

B. Huffman and O. Kuncar. Lifting and transfer: A modular design for quotients in isabelle/hol. In G. Gonthier and M. Norrish, editors, *Certified Programs and Proofs (CPP) 2013*, pages 131–146. Springer-Verlag, Berlin, 2013. 15

K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*. Springer-Verlag, New York, second edition, 1990. 5, 8, 16, 18, 19, 20, 23, 29, 30

J. Keränen. The identity problem for realist structuralism. *Philosophia Mathematica*, 9(3): 308–330, 2001. 13

S. Lang. *Algebraic Number Theory*. Springer-Verlag, New York, second edition, 1994. 18, 24, 29

S. Lang. *Algebra*. Springer-Verlag, New York, third edition, 2002. 5, 19, 20, 23, 28, 30

D. Lee, C. Szegedy, M. N. Rabe, S. M. Loos, and K. Bansal. Mathematical reasoning in latent space. In *International Conference on Learning Representations (ICLR) 2020*. OpenReview.net, 2020. 4

R. Y. Lewis and P. Madelaine. Simplifying casts and coercions (extended abstract). In P. Fontaine, K. Korovin, I. S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Practical Aspects of Automated Reasoning (PAAR) 2020*, volume 2752 of *CEUR Workshop Proceedings*, pages 53–62. CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2752/paper4.pdf`. 15

K. Manders. The Euclidean diagram. In P. Mancosu, editor, *The Philosophy of Mathematical Practice*, pages 80–133. Oxford University Press, Oxford, 2008. 28

N. D. Megill. Metamath. In F. Wiedijk, editor, *The Seventeen Provers of the World*, pages 88–95. Springer-Verlag, Berlin, 2006. 1

A. F. Monna. The concept of function in the 19th and 20th centuries, in particular with regard to the discussions between Baire, Borel and Lebesgue. *Archive for History of Exact Sciences*, 9 (1):57–84, 1972. 21

R. L. Morris. Motivated proofs: what they are, why they matter and how to write them. *Rev. Symb. Log.*, 13(1):23–46, 2020. 30

J. R. Munkres. *Topology*. Prentice Hall, Inc., Upper Saddle River, NJ, second edition, 2000. 16, 18, 30

T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*. Springer-Verlag, Berlin, 2002. 1

I. Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley & Sons, New York-Chichester-Brisbane, fourth edition, 1980. 8

C. Parsons. Structuralism and metaphysics: The foundations of mathematics and logic. *The Philosophical Quarterly*, 54(214):56–77, 2004. 11

A. Paskevich. *Méthodes de formalisation des connaissances et des raisonnements mathématiques: aspects appliqués et théoriques*. PhD thesis, Paris XII University, 2007. 2

E. H. Reck. Dedekind's structuralism: an interpretation and partial defense. *Synthese*, 137(3): 369–419, 2003. 11

M. D. Resnik. *Mathematics as a Science of Patterns*. The Clarendon Press, Oxford University Press, New York, 1997. 11

J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001. in two volumes. 4

H. L. Royden. *Real analysis*. Macmillan Publishing Company, New York, third edition, 1988. 5, 16, 18, 23, 24, 29

D. Selsam, S. Ullrich, and L. de Moura. Tabled typeclass resolution. *CoRR*, abs/2001.04301, 2020. URL `https://arxiv.org/abs/2001.04301`. 26

S. Shapiro. *Philosophy of Mathematics: Structure and Ontology*. Oxford University Press, New York, 1997. 11

M. Sozeau and N. Oury. First-class type classes. In O. A. Mohamed, C. A. Muñoz, and S. Tahar, editors, *Theorem Proving in Higher Order Logics (TPHOLs) 2008*, pages 278–293. Springer-Verlag, Berlin, 2008. 4, 26

E. M. Stein and R. Shakarchi. *Fourier Analysis: An Introduction*. Princeton University Press, Princeton, NJ, 2003. 19, 20

T. Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, Princeton, NJ, 2013. 15

P. Wadler and S. Blott. How to make ad-hoc polymorphism less ad-hoc. In *Principles of Programming Languages (POPL) 1989*, pages 60–76. ACM Press, 1989. 20

M. Wenzel. Type classes and overloading in higher-order logic. In E. L. Gunter and A. P. Felty, editors, *Theorem Proving in Higher Order Logics (TPHOLs) 1997*, pages 307–322. Springer-Verlag, Berlin, 1997. 4, 26

A. P. Youschkevitch. The concept of function up to the middle of the 19th century. *Archive for History of Exact Sciences*, 16(1):37–85, 1976/77. 21