

The Language of Mathematics

Mohan Ganesalingam

The accompanying thesis is part of a long-term project to enable computers to do mathematics in the same way that humans do. I will sketch something of the nature of mathematics and the project, and then turn to role of the thesis.

Mathematics

Mathematics arises from the interaction of two dissimilar modes of reasoning: a ‘soft’ side, dealing with ideas and analogies, and a ‘hard’ side, dealing with verification. The ‘hard’ side is easier to pin down. It consists primarily of formal ‘proofs’, each consisting of a series of assertions. A mathematician can verify that a proof is correct by following it, step by step, checking that each step follows from previous ones via facts already proved to be correct.

The ‘soft’ side is less easily described. It consists of intuitions about the formal objects constructed in mathematical proofs; ideas that one piece of mathematics may analogically correspond to another piece of mathematics; or even analogies between mathematics and objects in the physical world. For example, one proof of a famous theorem (The Fundamental Theorem of Algebra) is at heart based on the ‘soft’ idea that, if you place a rubber band on a flat surface and place your finger in the middle, then however you slide the rubber band around, it will remain wrapped around your finger. Another soft idea is that, if you want to show that two things are very similar to each other, it is sometimes easiest to work indirectly by showing that they are both very similar to a third thing.

The language which mathematicians use in textbooks and papers bridges the gap between these two modes of reasoning. It is hard and precise and presents rigorous proofs; but at the same time, it tries to subtly convey ephemeral, intangible soft ideas *inside* its hard constructs, via analogy, allusion or other indirect means. These soft ideas are rarely conveyed in a few words: one can seldom point at a single sentence and say, that sentence conveys all the ideas in this proof. But a precise, technical mathematical text can slowly build up a web of ephemeral, intangible concepts by careful choice of wording, drawing parallels between parts of mathematics, and similar means; and a human mathematician can read that text and be led surely to the underlying ideas. And, ultimately, it is these hidden soft ideas that matter. The hard content is important, because it is objective and verifiable; but the soft ideas are the point of mathematics, the things that mathematicians are ultimately searching for.

Because of its dual nature, the language of mathematics presents a remarkable opportunity for computers. Computers are very good at hard reasoning, and very bad at soft. The language of published mathematics, which captures soft inside hard, gives us a unique opportunity to make computers approach soft reasoning. Unfortunately, current computer approaches to mathematics, computer *theorem proving* programs, do not do anything of this kind. From a mathematician’s perspective, such programs have three deficiencies. First, interacting with them involves using impenetrable languages far removed from the normal language of mathematics, which (as we will see below) is essentially an augmented variety of English or another natural language.¹ Second, computers are very bad at mathematics — they cannot even check undergraduate-level proofs without extensive human assistance. Third, they *only* have the capacity to do hard mathematics; in the rare cases where they prove new results, they somehow manage to prove them without using any new soft ideas. Such soulless proofs are of interest to very few mathematicians.²

In my view, these three deficiencies are related. Theorem proving programs are ineffective at mathematics precisely because they have no capacity for soft ideas. And soft ideas are not explicitly taught, but implicitly absorbed from the actual, everyday, language used by mathematicians. Humans seem unable to extract such ideas from the alien, impoverished languages used by current theorem provers — so there is no reason to believe that computers could do so. The *actual*, hard language of mathematics gives us a very imperfect grip on the underlying soft ideas; but that is still the best grip we have.

For the reasons just described, the first three goals of my project are: to construct a computer language which is extremely close to the real language of mathematics, to write a body of mathematics in this language which conveys soft ideas to a real mathematician, and to enable a computer to extract some of those ideas — if not as well as a human, then still well enough to verify proofs written in the normal language of mathematics.

Language

The accompanying thesis is related to the first step in my project, the construction of a computer language. The difficulties that arise here have very

¹Cf. ‘What I would very much like to see is a more mathematician-friendly version of [the theorem proving language] Coq, where what you typed in was closer to what you would actually write as a mathematician and the computer translated it into the lower-level language for you.’ (Gowers, 2008).

²Cf. ‘We like our proofs to provide explanations rather than just formal guarantees of truth.’ (Gowers, 2000, p. 3).

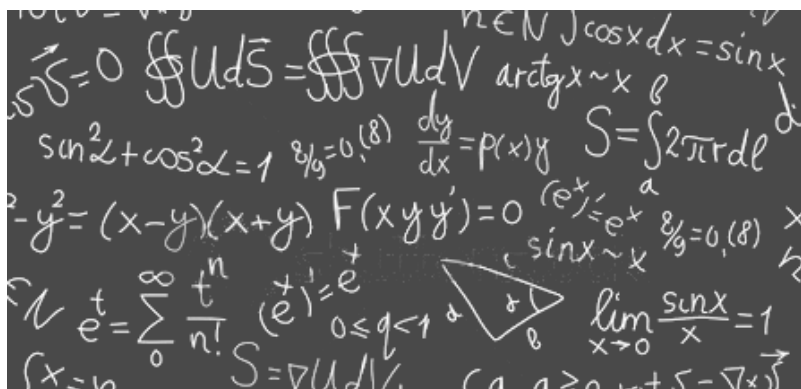
little to do with computers. Writing computer programs is easy; knowing which programs to write is hard. Before we can build a computer language *resembling real mathematical language*, we need to understand mathematical language extremely well; and we need not just the intuitive, subjective feel for mathematical language that mathematicians have, but a formal, objective, and above all precise analysis.

When looking for such an analysis, computer science and analyses of existing computer languages are of no help; the fully man-made languages analysed by computer scientists are exceptionally simple compared to the language of mathematics, which has developed organically over centuries. Our best hope is to turn to modern linguistics, which aims to lay bare the structures of rich, complex human languages. Mathematics differs sufficiently from human languages that little linguistic theory can be used directly; most of the material must be rebuilt and reconstructed. But linguistics, at the very least, gives us general ideas about how we can formally analyse language: it gives us a mindset and a starting place. Thus, in very broad terms, the thesis may be regarded as ‘the application of linguistics to mathematical language’. This is almost untrodden ground; the only linguist to analyse mathematics, Aarne Ranta, notes that

Amazingly little use has been made of [mathematical language] in linguistics; even the material presented below is just an application of results obtained within the standard linguistic fragment [...].

Ranta (1994)

Popular portrayals often present mathematics as being abstruse and incomprehensible, with a chaotic structure:



Thankfully, the reality is quite different. Mathematics is written in sentences and paragraphs, just like general language (Figure 1). The sentences contain a mixture of words and distinctively mathematical symbols. To understand mathematical language, I need to understand how the words function, how the symbols function, and how the two fit together. I analyse the words by

taking ideas from linguistics and adapting them extensively for mathematics. This part of my work is comparable to Ranta's analyses, although I analyse a much wider range of phenomena and overcome technical deficiencies noted by Ranta in his own work. I then show that the symbols in mathematics behave unlike anything anyone has ever studied in linguistics or in computer science, and create a new kind of theory to describe these. And finally, I develop ways to unify my dissimilar analyses of words and symbols to create a more general 'theory of mathematical language', different in character to any existing theory of natural or artificial languages.

If $K \leq G$ and there are inclusions $gKg^{-1} \leq K$ for every $g \in G$, then $K \triangleleft G$: replacing g by g^{-1} , we have the inclusion $g^{-1}Kg \leq K$, and this gives the reverse inclusion $K \leq gKg^{-1}$.

The kernel K of a homomorphism $f : G \rightarrow H$ is a normal subgroup: if $a \in K$, then $f(a) = 1$; if $g \in G$, then $f(gag^{-1}) = f(g)f(a)f(g^{-1}) = f(g)f(g^{-1}) = 1$, and so $gag^{-1} \in K$. Hence, $gKg^{-1} \leq K$ for all $g \in G$, and so $K \triangleleft G$. Conversely, we shall see later that every normal subgroup is the kernel of some homomorphism.

Figure 1: Mathematical language. Words are in black; symbols are in blue.

Even after constructing this unified theory, there is a great deal of work to do. If, like Ranta, one is considering only a small area of mathematics, most words and symbols have a single, fixed meaning. But for my purposes, I need a theory that can simultaneously describe all of undergraduate pure mathematics, and more. In this situation, a word like 'prime' can have a dozen meanings, and a simple expression like ' $x + y$ ' can mean a dozen different things depending on the context. In other words, *ambiguity* is a problem. I show that the amount of ambiguity increases exponentially as one considers more mathematics, and that the symbols in mathematics exhibit radical and novel kinds of ambiguity that have no parallel in any other kind of language. I also show that although ambiguity in words and ambiguity in symbols are entirely different in character, they are inextricably intertwined; neither can be analysed independently of the other. In order to give a proper linguistic analysis of mathematics, I need a theory that removes all of this ambiguity and predicts which meaning is intended in any given context.

I demonstrate that no linguistic techniques can remove the ambiguity in mathematics. I then borrow from mathematical logic and computer science a notion called the 'type' of an object, describing what kind of thing a mathematical object *is* and how it *behaves*, and show that this notion carries enough information to resolve ambiguity. However, I show that if we penetrate deeply enough into mathematical usage, we hit a startling and

unprecedented problem. In the language of mathematics, what an object *is* and how it *behaves* can sometimes differ: numbers sometimes behave as if they were not numbers, and objects that are provably not numbers sometimes behave like numbers. So the two components of the notion of type slide out of alignment with each other, in an unprecedented way. Ultimately I show that the standard concept of ‘type’ is tangled, and tease out two distinct notions which clarify the situation.

Even after doing this, systematically removing ambiguity by *deducing* which types are present in a given sentence is a major undertaking. The behaviour of symbols in mathematics is much more complex than in computer languages, so methods used to deduce types in computer science are inapplicable. Additionally, no one has ever used a notion of type to disambiguate words and sentences, let alone unified that method with any method of disambiguating symbolic material. Ultimately, I fuse existing notions of type with much more linguistic notions, and by doing so develop a method that can remove all ambiguity from mathematical sentences. Unlike many linguistic methods, it applies no guesswork; the answers it gives are guaranteed to be correct.

Finally, I turn to some problems in the underlying mathematical material itself. I show that in the ‘foundations’ of mathematics, which treat simple objects like numbers, there is a considerable gap between what mathematicians claim is true and what they believe, and that this mismatch causes linguistic problems. For example, mathematicians claim that all numbers are really objects of a different kind, so-called ‘sets’, but their use of language consistently reflects a belief that numbers are not sets. I show that this issue relates to a famous problem in the philosophy of mathematics, raised in the seminal *What Numbers Could not Be* (Benacerraf, 1965). I then introduce a novel notion of time, centred on the idea that mathematics is *learnt incrementally* by individual mathematicians, and use this to show that all of the standard accounts of the foundations of mathematics are flawed: some predict that no one could ever learn mathematics, and others require that no new mathematics ever be discovered. I then construct a new account of the foundations which is in accord with mathematical intuitions while satisfying all of the philosophical and mathematical constraints, and rely on this to remove the linguistic problems.

Future Research

This thesis completes my linguistic analysis of mathematics. The next step in my project concentrates on texts. I have chosen a particularly self-contained area of mathematics, called group theory, and am copying material on that subject from a standard textbook into my computer language.

When this process is complete, the resulting text will contain all the group theory that an undergraduate mathematician learns. Because the language is so close to real mathematics, I can ensure that human mathematicians can still see the soft ideas in the text; indeed, the ability to convey such ideas *to humans* set the standard for my linguistic analysis and language construction.

Once I complete the text, I will analyse the material in it like a mathematician, but working more intensively; I will examine every step of every proof and try to understand the ideas motivating and justifying that step in detail. And whenever I find an idea, however small, in this way, I will try and root that idea in the text: I will search for tiny textual clues or parallels between arguments that might account for the genesis of the idea, or its use in a particular context. Eventually, when I understand the underlying processes well enough — and not before — I will try to replicate them on computer. In doing so, I will rely absolutely on the fact that the computer can understand the text in a ‘hard’ way, because of the work presented in this thesis.

References

- P. Benacerraf (1965). ‘What numbers could not be’. *The Philosophical Review* **74**(1):47–73.
- W. T. Gowers (2000). ‘Rough structure and classification’. *Geometric and Functional Analysis, Special Volume: GAFA2000 ‘Visions in Mathematics’ (Tel Aviv, 1999), Part I*, pp. 79–117.
- W. T. Gowers (2008). ‘Gowers’ weblog’. <http://gowers.wordpress.com>, entry for 28 July 2008.
- A. Ranta (1994). ‘Type theory and the informal language of mathematics’. In H. Barendregt & T. Nipkow (eds.), *Proceedings of the 1993 types workshop, Nijmegen, Lecture Notes in Computer Science*, vol. 806, pp. 352–365. Springer-Verlag.