Jahresber Dtsch Math-Ver (2021) 123:3–26
https://doi.org/10.1365/s13291-020-00221-1

SURVEY ARTICLE

Formalising Mathematics – in Praxis; A Mathematician's First Experiences with Isabelle/HOL and the Why and How of Getting Started

Angeliki Koutsoukou-Argyraki[1]

Published online: 6 October 2020

Abstract This is an account of a mathematician's first experiences with the proof assistant (interactive theorem prover) Isabelle/HOL, including a discussion on the rationale behind formalising mathematics and the choice of Isabelle/HOL in particular, some instructions for new users, some technical and conceptual observations focussing on some of the first difficulties encountered, and some thoughts on the use and potential of proof assistants for mathematics.

Keywords Interactive theorem proving · Isabelle/HOL · Proof assistant · Formalisation of mathematics

Mathematics Subject Classification (2020) 03B35 · 68V15 · 68V20 · 68V35

"...We believe that when later generations look back at the development of mathematics one will recognise four important steps: (1) the Egyptian-Babylonian-Chinese phase, in which correct computations were made, without proofs; (2) the ancient Greeks with the development of 'proof'; (3) the end of the nineteenth century when mathematics became 'rigorous'; (4) the present, when mathematics (supported by computer) finally becomes fully precise and fully transparent." Barendregt, H. and Wiedijk, F. (The challenge of computer mathematics, Philos. Trans. - Royal Soc., Math. Phys. Eng. Sci. 36(1835):2351–2375 (2005) [4]).

1 Introduction

Writing a mathematical proof could be compared to making a beautiful carpet featuring elaborate patterns: the end result looks impressive, mind-boggling even, but

✉ A. Koutsoukou-Argyraki
ak2110@cam.ac.uk

[1] Department of Computer Science and Technology, University of Cambridge, Cambridge, UK

the actual process in fact consists of simple arguments, like small stitches. While formalising[1] mathematics with a proof assistant (also referred to as interactive theorem prover), that is, writing proofs in a formal language so that a computer can verify the correctness of a proof, every simple argument should be made explicit: it can be a painstaking process, as every step that is trivial to a human may need to be analysed in a number of elementary sub-steps. But this process could be invaluable to students (and sometimes even to professional mathematicians) who can gain a deeper understanding of a proof. This is of course far from the only motivation to formalise mathematics with a proof assistant: verification of proofs, as well as contributing to the constantly growing libraries of formal proofs that can be used for creating tools to assist mathematicians in their creative work, are also very important motives (a discussion on this will follow in Sect. 2).

It should be clarified that proof assistants are not the same as computer algebra tools, such as MATLAB or Mathematica: the latter mainly do computations, while the former mainly do *reasoning*. Proof assistants can be used to verify various algorithms that may have many different applications in theoretical computer science and in industry too, but here we discuss the use of proof assistants for formalising and verifying *mathematical* proofs in particular. Various different proof assistants that are based on different formal systems have been developed over the years. A comprehensive review comparing different theorem provers (as of 2006) edited by Freek Wiedijk and forwarded by Dana S. Scott is a useful reference for interested users [65]; seventeen systems[2] are showcased by presentation and direct comparison of formalised proofs. Since then, the state-of-the-art has improved with libraries of formal proofs having expanded significantly and with the new proof assistant Lean having entered the picture as well.

In an online list [66], Freek Wiedijk is keeping track of which theorems, out of a list of 100 significant mathematical theorems, have been formalised in which proof assistant, considering ten[3] different systems. As of today, 95 out of the 100 theorems of the list have been formalised in at least one of these ten systems, and Isabelle comes second after HOL Light in the list with 83 out of the 100 theorems having been formalised in Isabelle.

Nowadays, the most popular proof assistants for formalising mathematics are Isabelle, Coq, Lean, Mizar, HOL4, HOL Light, Agda and Metamath, but this article, as the title suggests, focusses on Isabelle/HOL (: Higher Order Logic) in particular. Isabelle was first developed by Lawrence Paulson and Tobias Nipkow [46, 51] and supports different logical formalisms, such as first and higher-order logic as well as Zermelo Fraenkel set theory (Isabelle/ZF). Isabelle/ZF and Isabelle/HOL share the same basic inference procedures and user interface, however Isabelle/HOL is more developed and, as we will see, features useful automation tools.

[1] I consider the terms "mechanising"/ "mechanisation" as perhaps more appropriate than the terms "formalising"/ "formalisation" in our context, as the latter may be confused with formalisation in the sense of what was attemped e.g. in Principia Mathematica by Whitehead and Russell [62]. Nevertheless, I use "mechanisation" and "formalisation" interchangeably, as it is usually done in the literature; in recent years, these terms are being used as synonymous.

[2] HOL, Mizar, PVS, Coq, Otter/Ivy, Isabelle/Isar, Alfa/Agda, ACL2, PhoX, IMPS, Metamath, Theorema, Leog, Nuprl, Omega, B method and Minlog.

[3] HOL Light, Isabelle, Metamath, Coq, Mizar, ProofPower, Lean, PVS, nqthm/ACL2, NuPRL/MetaPRL.

Proof assistants are becoming more and more popular among younger mathematicians and students. For example, note the prize-winning project by a group of undergraduate students from Bremen formalising Matiyasevich's proof of Hilbert's 10th problem with Isabelle/HOL [54, 58], as well as the vibrant community of young mathematicians and undergraduate students working under the direction of Kevin Buzzard at Imperial College London with Lean [7]. This enthusiasm has in part been fuelled by several important milestones, such as the verification of the proof of the four-colour theorem in Coq by Georges Gonthier [12], the verification of the proof of the Kepler conjecture [16] in HOL Light and Isabelle by Thomas Hales et al. [17] as well as the formalisation of Gödel's incompleteness theorems using Isabelle/HOL by Lawrence Paulson [48, 49].

Hoping that this may be of help to students interested in formalisation of mathematics and automated reasoning, I am sharing this report[4] on my comments (mostly focussing on the difficulties) on my experience with Isabelle/HOL working within the ALEXANDRIA project at the University of Cambridge, as a pure mathematician who had no prior formalisation experience. My goal is to give an overview of my experience with Isabelle/HOL so far and of my views on the formalisation of mathematics, and to share some instructions on getting started with Isabelle as well as some first observations which I made, both technical and conceptual, that might prove to be helpful to students and beginners in their early learning stages.

The plan of this paper is as follows: in Sect. 2, I discuss the main motives for formalising mathematics also commenting on my own motivation and research background and I introduce the ALEXANDRIA project at the University of Cambridge. In Sect. 3, I give a brief introduction on how to get started with Isabelle/HOL. In Sect. 4, I discuss the first difficulties I encountered with Isabelle/HOL. In Sect. 5, I give an overview of the first projects I worked on as case studies to explore what Isabelle/HOL can and cannot do. Section 6 is to be read as a caution to new users showing how to avoid making a bad use of a proof assistant, in particular I give examples of different kinds of mathematical mistakes and show how Isabelle would not detect them if the user is not making a responsible use of it. Finally, Sect. 7 involves a comment on the main two different aspects of revolutionising mathematical practice through the mechanisation of mathematics.

2 Motivation for Formalising Mathematics and the ALEXANDRIA Project

2.1 Why Formalise Mathematics?

However one would like to describe mathematical practice, it is certainly not the same as programming (nor should it be reduced to it). As computers and artificial intelligence are becoming more and more integrated into every aspect of our lives, "modernising" mathematical practice in that respect does not sound unexpected, but this is not an issue of modernisation for the sake of it (although for some people this

[4] This is an updated version of some earlier online notes from July 2019 that I had shared on ResearchGate.

would be a sufficient reason); in fact, formalising mathematics offers many direct benefits. There are different approaches to making a choice about what material to formalise. These are: (a) formalising the mathematical curriculum, that is, basic material that undergraduate students are usually taught; (b) formalising advanced, famous results, e.g. the aforementioned formalisations of Gödel's incompleteness theorems, the four-colour theorem and Hilbert's 10th problem; (c) formalising new research results, that can be either from mainstream research papers or that could be considered groundbreaking (like the aforementioned proof of the Kepler conjecture). To the above we can add a fourth strategy, which is not quite achievable with the current state-of-the-art but is one of the main goals for the future of the field: (d) discover new mathematical results through the process of formalising.[5] To this end, advances in machine learning are widely regarded as necessary. Timothy Gowers has described how an interactive assistant would ideally "converse" with a human mathematician to assist the discovery of new results [14]. But, today, we are still far from this goal.

There are different expected benefits that motivate each strategy.

An obvious first reason is verification. This applies mostly to (c), since in all other cases the material has been checked by a great number of people over the years and we wouldn't expect to find mistakes in elementary material. With more advanced and more recent results the probability of finding mistakes of course starts to increase. A recent example in the theory of Gromov-hyperbolic spaces is a mistake found through formalisation in Isabelle by Gouëzel and Shchur [13].

Another reason, applying mostly to (a) and (b), is that contributing to the (very fast growing) libraries of formal proofs amounts to the creation of a database with a huge potential. A "physical" mathematical library consisting of "material" books, or even an online library consisting of pdf files, is more restrictive – while here we have to do with a library written in code: something we can modify, interact with, reuse. More importantly, formalised material can be used to create tools for goal (d). A vision for the future is the creation of an interactive assistant that would provide "brainstorming" tips to research mathematicians in real time assisting them in the process of discovering (or inventing) a new result.

A third reason, applying to all (a), (b), (c) (and (d)), is that the process of formalising in itself can help the user gain brand new insights even in already familiar topics. To a large extent this is because of the high level of detail in which a formalised proof must be written, but also because using new tools forces to look at familiar material from a new angle.[6]

Last but not least, formalisation can also serve educational purposes (this applies mostly to (a)).

I will not elaborate further on a general discussion about this topic – I rather restrict to mentioning the inspiring papers by Avigad [3], Paulson [50], Lamport [42], Barendregt and Wiedijk [4] and the classical QED Manifesto from the mid-90's [6].

[5] Although the original proofs of the four-colour theorem and the Kepler conjecture were computer-assisted, they were not discovered through the process of formalising, but were formalised later for verification purposes.

[6] Kevin Buzzard gave an instance of this phenomenon regarding notions of equivalence and similarity for groups during his talk at *Big Proof* which took place in May 2019 at the International Center for Mathematical Sciences in Edinburgh https://www.icms.org.uk/bigproof.php.

Pioneering mathematicians such as the Fields medalists Timothy Gowers [14] and Vladimir Voevodsky [61] and more recently Kevin Buzzard [7–9, 11], Peter Koepke [24] and Benedikt Löwe [43], among others, have been very strong advocates for the use of proof assistants and formal verification in mathematics.

2.2 More Personal Motivation and My Mathematics Background

As a pure mathematician with some background in logic and proof theory, my own interest in the topic was initially driven not only by a fascination for the emerging culture of re-imagining mathematical practice in the light of new AI developments but also by philosophical questions on the nature of mathematical proofs, e.g. when encountering in my own research work different proofs of similar statements giving completely different computational content [29, 31].

In particular, regarding my mathematics background, my PhD research [32] was pen-and-paper work and involved applications of proof theory to mathematics (mainly in nonlinear analysis) [28–31]. I have been working within Ulrich Kohlenbach's proof mining programme [25–27] that involves pen-and-paper extraction of constructive/quantitative information from proofs in the form of computable bounds, which requires a logical analysis of a proof and rewriting it to make the logical form of all the statements involved explicit via revealing the hidden quantifiers. It is only natural that such a line of research would provoke the question:

What is it that makes a "good" proof?

which of course has many possible answers:

- a shorter proof;
- a more "elegant" proof (which is of course usually subjective);
- a simpler proof (consider Hilbert's 24th problem (1900): *"find criteria for simplicity of proofs, or, to show that certain proofs are simpler than any others." [45]*);
- thinking in terms of Reverse Mathematics – a proof in a weaker subsystem of Z_2 (Second Order Arithmetic);
- an interdisciplinary proof (e.g. a geometric proof for an algebraic problem or vice-versa would be considered to give a deeper mathematical insight);
- a proof that is easier to reuse i.e. if it provides some algorithm or technique or intermediate result that can be useful in different contexts too;
- a proof giving "better" computational content.

In the aforementioned proof mining papers [29, 31] it turns out that different proofs of similar statements (about the fixed points of nonexpansive semigroups) give completely different computational content, which in turn, provokes the question:

What do we mean by "better" computational content?

- a bound of lower complexity?
- a bound that is more precise numerically?
- a bound that is more "elegant"?

It would be reasonable to ask too:

- How are the aforementioned proof features related to each other (if at all)?

- Could we ever ensure that we get the optimal computational content (from a given proof)?

The answers to the above questions are non-trivial – and my hope is that formalisation could provide us with some useful insights.

I joined the ALEXANDRIA project in October 2017, some months after obtaining my PhD in pure mathematics and with no prior formalisation experience in any proof assistant. Although proof mining has trained me in patiently de-constructing proofs into every elementary sub-step and in tidying up proofs after finding their underlying logical structure, which are both necessary skills for anyone interested in formalisation, starting to explore a proof assistant was to me a brand new challenge.

2.3 The ALEXANDRIA Project

The project ALEXANDRIA: *Large-Scale Formal Proof for the Working Mathematician* led by Lawrence Paulson and funded by the European Research Council, started at the University of Cambridge in Autumn 2017 and is aiming at contributing to the creation of a proof development environment for working mathematicians through a collaboration of mathematicians and computer scientists. The focus of the project is the management and use of large-scale mathematical knowledge, both as theorems and as algorithms. To formalise mathematical proofs we make use of the proof assistant Isabelle/HOL. Our goals involve not only the contribution to the Isabelle libraries with new formalised material, but also the management and organisation of the libraries, as well as the creation of interactive tools that would assist mathematicians in the process of both formalising known proofs and discovering new results. To this end, tools facilitating automation and search are important priorities. The ALEXANDRIA group members are Lawrence Paulson, Wenda Li, Anthony Bordg, Yiannos Stathopoulos and myself. For more information I refer to the project description by Lawrence Paulson [47].

3 First Steps in Isabelle/HOL

Isabelle has several major advantages over other proof assistants: first of all, it uses the structured language *Isar*, which is understandable by both humans and machines. Moreover, it supports automation tools, which are much more efficient than automation in any other proof assistant. The main automation tool featured by Isabelle/HOL is *Sledgehammer* [5], which searches for proofs by calling a number of external automated theorem provers, and afterwords converts the proofs (if found) to Isar. Also, there is the command `try0`, which calls a number of widely used proof methods in search of a proof (`simp`, `auto`, `blast`, `metis`, `argo`, `linarith`, `presburger`, `algebra`, `fastforce`, `force`, `fast`, `meson`, `satx`). There are also certain counterexample finding tools (*nitpick* and *Quickcheck*). Sledgehammer, nitpick and Quickcheck are not supported by Isabelle/ZF. We will discuss more Isabelle's automation later. Another advantage of Isabelle is that its libraries are already quite extensive – a great deal of material has been formalised over the years and the libraries are constantly growing. Isabelle is based on simple types and the internal/implementation languages are Standard ML and Scala. It does admit classical (i.e. non-constructive) proofs. In particular, Isabelle/HOL includes the axiom of

choice. The default user interface is Isabelle/jEdit, which provides real-time proof checking, as well as rich semantic information for the formal text and direct links to the user manuals and tutorials[7] (see "Documentation" at the left-hand side of the user interface).

The first step is of course to download Isabelle (which is distributed for free under open-source licenses) from the Isabelle website [21]. Installation is straightforward. Next, one should get familiar with the Isabelle Libraries and the Archive of Formal Proofs (AFP) [1]. The HOL and ZF Libraries can be found in [22] and [23] respectively. The AFP contains a vast collection of formalised material in a variety of topics in mathematics (algebra, analysis, probability theory, number theory, geometry, topology, graph theory, combinatorics, category theory) as well as some applications in economics and physics. Moreover, the AFP contains extensive proof libraries in computer science and logic. As of 2 September 2020, there are 553 entries by 362 authors [1]. Isabelle users can get help with their questions on the Isabelle mailing list[8] as well as on the Isabelle Zulip chat.[9]

Reading the manuals and tutorials can be helpful, but usually learning-by-doing can be more efficient. It would be thus advisable to start by exploring some theories from the library (e.g., after having installed Isabelle, by opening a session and going to

File→ Open→ isabelle→ src→ HOL→ Analysis

to get to the Analysis Library theory files) and/or download one of the AFP entries (there are detailed instructions on the website [1]) and make modifications to see how it responds.

A theory file starts with `theory` [the name of the file] followed by `imports` [the libraries or AFP entries that need to be imported for the purposes of the theory at hand]. The theory is then developed between the keywords `begin` and `end`. The file should be saved with the same name that is used right after `theory`, followed by ".thy".

The user can call the automation tools to look for proofs simply by typing `try0` and `Sledgehammer`. In order to search for already formalised material in the loaded theories of the active session, the user can type `find_theorems` or `find_theorems name:` followed by some search word. While looking at an open .thy file, the user can place the cursor on any object and by pressing the Command key and clicking, the jEdit interface takes the user to the theory where the object is defined. The keyword `sorry` can be used in the place of a missing proof, so that a statement can be temporarily regarded as "proven" and the user may even refer to the result to use it in other proofs and return to the missing proof later. The keyword `oops` has a similar use, except that it "abandons" a proof typically in the middle; unlike `sorry`, it cancels the entire proof attempt up to the previous `lemma`/`theorem`/`proposition`/`corollary` keyword and the claimed statement is not available to be used.

[7] These can also be found under "Documentation" on [21].

[8] https://lists.cam.ac.uk/mailman/listinfo/cl-isabelle-users.

[9] https://isabelle.zulipchat.com.

Looking at examples online from the Library [22] and the AFP [1], the user will notice the variation of colours, which helps to distinguish between different kinds of keywords. For example, text with comments or headings, and facts in quotations that had been previously assumed/defined/shown within the same proof appear in orange, words indicating the structure of a theory or a proof, like `theorem`, `lemma`, `proof`, `by`, `qed`, `subsection` appear in navy blue, keywords of an imperative nature within a proof such as `fix`, `assume`, `thus`, `show`, `obtain` appear in light blue, keywords indicating the posing of assumptions and conclusions i.e. `assumes`, `fixes`, `and`, `where`, `shows`, `if`, `is` appear in green, entire mathematical statements appear in pink, the keywords `apply` and `done` referring to applying one or more proof methods appear in red, comments within the symbols (* and *) appear in dark red, and everything else appears in black.

Note that while working directly on a formalisation within an active Isabelle session the colours will differ, namely the user will notice that fixed/free variables appear in blue and bound variables appear in green.

3.1 Examples

Most new users would agree that Isar looks understandable, at least not much less than LaTeX code: a few examples with some clarifications are given below. The reader is invited to locate the following examples, which can be accessed online on the Library or the AFP as given. However, in order to be able to work directly with the theory (and to decipher any references to unknown objects by Command-Click which will take the user to the wanted definition of the object in question), the user must open these theories via an Isabelle session following the previously given instructions.

3.1.1

A definition of the 7-dimensional real cross product from my AFP entry on Octonions [34]: We have defined a multiplication operation between two 7-dimensional real vectors $\vec{a}$ and $\vec{b}$ giving a 7-dimensional real vector, so that each of its coordinates is given in terms of the coordinates of $\vec{a}$ and $\vec{b}$; for example, the first coordinate will be given by $a_2 \cdot b_4 - a_4 \cdot b_2 + a_3 \cdot b_7 - a_7 \cdot b_3 + a_5 \cdot b_6 - a_6 \cdot b_5$.

```
definition cross7 :: "[real^7, real^7] ⇒ real^7"  (infixr "×₇" 80)
  where "a ×₇ b ≡
    vector [a$2 * b$4 - a$4 * b$2 +  a$3 * b$7 - a$7 * b$3  +  a$5 * b$6 - a$6 * b$5 ,
            a$3 * b$5 - a$5 * b$3 +  a$4 * b$1 - a$1 * b$4  +  a$6 * b$7 - a$7 * b$6 ,
            a$4 * b$6 - a$6 * b$4 +  a$5 * b$2 - a$2 * b$5  +  a$7 * b$1 - a$1 * b$7 ,
            a$5 * b$7 - a$7 * b$5 +  a$6 * b$3 - a$3 * b$6  +  a$1 * b$2 - a$2 * b$1 ,
            a$6 * b$1 - a$1 * b$6 +  a$7 * b$4 - a$4 * b$7  +  a$2 * b$3 - a$3 * b$2 ,
            a$7 * b$2 - a$2 * b$7 +  a$1 * b$5 - a$5 * b$1  +  a$3 * b$4 - a$4 * b$3 ,
            a$1 * b$3 - a$3 * b$1 +  a$2 * b$6 - a$6 * b$2  +  a$4 * b$5 - a$5 * b$4 ]"
```

3.1.2

A statement attesting that norm equality implies inner product equality (and the converse) together with a proof, from the Analysis/HOL Library [22], in particular from the theory Inner_Product by Brian Huffman: Notice that the proof here is "apply-

style" and not a structured Isar proof. The user can see the definitions of each symbol and of `norm`, `inner`, as well as the statements of the lemmas `order_eq_iff`, `norm_le` that are used in the proof by placing the cursor on the object in question, pressing the Command key and clicking.

```
lemma norm_eq: "norm x = norm y ⟷ inner x x = inner y y"
  apply (subst order_eq_iff)
  apply (auto simp: norm_le)
  done
```

3.1.3

Let us now look at an example in more detail, comparing the "informal" with the formal proof. Consider the following version of the Cauchy-Schwarz inequality together with a proof. An "informal" proof is given below. Note that we use [___] to denote the "trivial" intermediate arguments which are missing and will be explicitly shown in the formal proof.

Cauchy-Schwarz inequality: For every two vectors $\vec{x}$, $\vec{y}$ of an inner product space over the field of real numbers, $|\langle\vec{x},\vec{y}\rangle|^2 \leq \langle\vec{x},\vec{x}\rangle \cdot \langle\vec{y},\vec{y}\rangle$.

Proof: We distinguish two cases: The case that $\vec{y} = 0$ (in which the proof is trivial [___]) and the case that $\vec{y} \neq 0$. Assume that $\vec{y} \neq 0$. Let $r := \langle\vec{x},\vec{y}\rangle / \langle\vec{y},\vec{y}\rangle$. We have that: $0 \leq \langle\vec{x} - r\vec{y}, \vec{x} - r\vec{y}\rangle$ [___] $= \langle\vec{x},\vec{x}\rangle - \langle\vec{y},\vec{x}\rangle \cdot r$ [___] $= \langle\vec{x},\vec{x}\rangle - |\langle\vec{x},\vec{y}\rangle|^2/\langle\vec{y},\vec{y}\rangle$ [___]. So we have shown that $0 \leq \langle\vec{x},\vec{x}\rangle - |\langle\vec{x},\vec{y}\rangle|^2/\langle\vec{y},\vec{y}\rangle$ from which follows that $|\langle\vec{x},\vec{y}\rangle|^2/\langle\vec{y},\vec{y}\rangle \leq \langle\vec{x},\vec{x}\rangle$ [___] from which the wanted inequality immediately follows [___]. QED.

The above proof is written in Isabelle (this is from the Analysis/HOL Library [22], in particular from the theory Inner_Product by Brian Huffman) as follows (note that in the place of every [___] we have provided proofs which are indicated by the keyword `by` in navy blue followed by some proof method, or by some proof method and one or more lemmas):

```
lemma Cauchy_Schwarz_ineq:
  "(inner x y)² ≤ inner x x * inner y y"
proof (cases)
  assume "y = 0"
  thus ?thesis by simp
next
  assume y: "y ≠ 0"
  let ?r = "inner x y / inner y y"
  have "0 ≤ inner (x - scaleR ?r y) (x - scaleR ?r y)"
    by (rule inner_ge_zero)
  also have "... = inner x x - inner y x * ?r"
    by (simp add: inner_diff)
  also have "... = inner x x - (inner x y)² / inner y y"
    by (simp add: power2_eq_square inner_commute)
  finally have "0 ≤ inner x x - (inner x y)² / inner y y" .
  hence "(inner x y)² / inner y y ≤ inner x x"
    by (simp add: le_diff_eq)
  thus "(inner x y)² ≤ inner x x * inner y y"
    by (simp add: pos_divide_le_eq y)
qed
```

We continue with a few more examples of well-known statements formulated in Isabelle.

3.1.4

The statement of the Stone-Weierstrass theorem for polynomial functions, from the Analysis/HOL Library [22], in particular from the theory Weierstrass_Theorems by Lawrence Paulson:

```
theorem Stone_Weierstrass_polynomial_function:
  fixes f :: "'a::euclidean_space ⇒ 'b::euclidean_space"
  assumes S: "compact S"
      and f: "continuous_on S f"
      and e: "0 < e"
    shows "∃g. polynomial_function g ∧ (∀x ∈ S. norm(f x - g x) < e)"
```

3.1.5

A version of Zorn's lemma, together with a proof, from the HOL Library [22], in particular from the theory Zorn by Jacques Fleuriot, Tobias Nipkow and Christian Sternagel, ported from Lawrence Paulson's Zorn theory development from Isabelle/ZF:

```
subsubsection ‹Zorn's lemma›

text ‹If every chain has an upper bound, then there is a maximal set.›
theorem subset_Zorn:
  assumes "⋀C. subset.chain A C ⟹ ∃U∈A. ∀X∈C. X ⊆ U"
  shows "∃M∈A. ∀X∈A. M ⊆ X ⟶ X = M"
proof -
  from subset.Hausdorff [of A] obtain M where "subset.maxchain A M" ..
  then have "subset.chain A M"
    by (rule subset.maxchain_imp_chain)
  with assms obtain Y where "Y ∈ A" and "∀X∈M. X ⊆ Y"
    by blast
  moreover have "∀X∈A. Y ⊆ X ⟶ Y = X"
  proof (intro ballI impI)
    fix X
    assume "X ∈ A" and "Y ⊆ X"
    show "Y = X"
    proof (rule ccontr)
      assume "¬ ?thesis"
      with ‹Y ⊆ X› have "¬ X ⊆ Y" by blast
      from subset.chain_extend [OF ‹subset.chain A M› ‹X ∈ A›] and ‹∀X∈M. X ⊆ Y›
      have "subset.chain A ({X} ∪ M)"
        using ‹Y ⊆ X› by auto
      moreover have "M ⊂ {X} ∪ M"
        using ‹∀X∈M. X ⊆ Y› and ‹¬ X ⊆ Y› by auto
      ultimately show False
        using ‹subset.maxchain A M› by (auto simp: subset.maxchain_def)
    qed
  qed
  ultimately show ?thesis by blast
qed
```

3.1.6

The statement and a proof of the Riemann mapping theorem, from the Complex_Analysis/HOL Library [22], in particular from the theory Riemann_Mapping by Lawrence Paulson (note that the formal proof here is very short as it directly follows from auxiliary statements formalised in the same theory):

```
theorem Riemann_mapping_theorem:
    "open S ∧ simply_connected S ⟷
     S = {} ∨ S = UNIV ∨
     (∃f g. f holomorphic_on S ∧ g holomorphic_on ball 0 1 ∧
           (∀z ∈ S. f z ∈ ball 0 1 ∧ g(f z) = z) ∧
           (∀z ∈ ball 0 1. g z ∈ S ∧ f(g z) = z))"
    (is "_ = ?rhs")
proof -
  have "simply_connected S ⟷ ?rhs" if "open S"
    by (simp add: simply_connected_eq_biholomorphic_to_disc that)
  moreover have "open S" if "?rhs"
  proof -
    { fix f g
      assume g: "g holomorphic_on ball 0 1" "∀z∈ball 0 1. g z ∈ S ∧ f (g z) = z"
        and "∀z∈S. cmod (f z) < 1 ∧ g (f z) = z"
      then have "S = g ` (ball 0 1)"
        by (force simp:)
      then have "open S"
        by (metis open_ball g inj_on_def open_mapping_thm3)
    }
    with that show "open S" by auto
  qed
  ultimately show ?thesis by metis
qed
```

4 First Difficulties Encountered

4.1 Syntax

As in any new programming (or even natural) language, the first challenge for a new user is familiarisation with the syntax and the essential keywords. In fact, I found Isar to be both quite intuitive in terms of structure and easily readable, while the jEdit user interface is very user-friendly. The fact that Isar admits structured proofs is a major advantage. There are, however, certain (mostly syntactic) features that may seem surprising to a new user. Some miscellaneous characteristic examples are:

- The standard proof patterns
  `have "a<b" also have "...<c" finally show "a<c" by auto` and
  `have "a<b" moreover have "...<c" ultimately show "a<c" by auto`.
- The syntax does not allow for shortcuts such as "$a < b < c$" or "$a, b, c > 0$", instead one has to write `"a<b∧b<c"` or `"a<b" and "b<c"`, respectively `"a>0∧b>0∧c>0"` or `"a>0" and "b>0" and "c>0"`.
- The user has to remember to always include type information. Even arabic numbers are in certain cases regarded as constants of some unknown type unless their type is explicitly stated (e.g. `(1::int)/(2::int)^0 = 1` is proven just by the proof method `simp` while for `1/2^0 = 1` automation gives no answer nor do we get any explanatory error messages). Moreover, for exponentiation, if the exponent is of type real or integer one has to use `powr`, while for type natural the symbol `^` works. Also, often the user has to switch the type of a variable or constant (`of_int`, `of_real`, `of_nat`) when confronted for instance with division.
- The meaning of some keywords, e.g.: `where`, `that`, `when`, `at_top`, `sequentially`.

- The use of several symbols, e.g.: the meet and join operators for lattices are symbolised by $\sqcap$, $\sqcup$ instead of $\wedge$, $\vee$ that are normally used in the literature; different kinds of arrows; the absolute value symbol.
- Overall the extremely high level of detail in which proofs must be written.

Discovering such features for the first time may be slightly time-consuming for a new user as error messages are not always helpful. New users should not be discouraged by the potentially slow progress during the very first stages of their learning as after some practice the learning process accelerates significantly and becomes very rewarding.

4.2 Search Features

After syntax, search is of very high significance for the mathematician user, both in terms of finding material by thematic classification, and of finding essential technical lemmas and definitions while in the process of formalising new work. In the latter case, it is extremely helpful that the interface allows for words to function as hyperlinks: as mentioned, by pressing the Command key while clicking on each object, the user is taken to the theory where the object at hand is defined. However, there is room for improvement with respect to search features, from different points of view. In particular:

(1) It is possible to search for theorems only on the basis of symbolic patterns occurring in them or of their names (via `find_theorems` or `find_theorems name:` as mentioned) so an unexpected name could create obstacles. Even though `find_theorems` is helpful most of the time, in some cases it could be limiting. In particular:

- As search is done based on pattern-matching, especially new users may sometimes not know what are the appropriate search words for the notions that they are looking for (e.g. “summable” yields many results while “summability” yields no results, “infimum” and “supremum” yield no results while a lot of related material exists in the libraries etc). That is, searching for *associated concepts* would be more helpful.
- `find_theorems` is case-sensitive (e.g. “borel” gives 510 results while “Borel” gives no results). We are aiming for case-insensitive search.
- Search is performed only in the libraries and theories that have been already loaded by the user. Ideally, search would be performed in all the libraries (and the AFP) regardless of what the user has loaded.
- It would be useful to differentiate the search for facts based on mathematical objects related to their statements from the search for facts based on mathematical objects related to their proofs and to patterns that may occur in their proofs.
- An efficient method of filtering and ranking the search results would also be useful to have.

(2) At the same time, manual search in the Library can be rather time-consuming, which is a consequence of the four following facts: (a) the fast growing size of the Li-

brary, especially the Analysis Library. Tobias Nipkow has recently initiated an effort to create a comprehensive manual for the contents of the Analysis Library. Fabian Immler, Lawrence Paulson, Manuel Eberl, Wenda Li, Mohammad Abdulaziz and myself have been contributing to this work in progress. (b) The general difficulty in classifying mathematical knowledge, as very often the borders between mathematical disciplines are unclear. Users may have difficulty finding material that is classifiable in different ways. (c) It is not unusual in mathematical literature to use different names in different contexts for the same notion (*synonymy*). For instance, I had been looking for a formalisation of the valuation operator v_2 i.e. $v_2(n) = \max A$ for $n \in \mathbb{Z}$ where $A = \{k \in \mathbb{N} : 2^k \text{ divides } n\}$. I was informed by Manuel Eberl that he had formalised this under the name "multiplicity" (and in fact for p-order, not just 2-order), in the Computational Algebra Library, in Factorial.Ring.thy [22]. (d) And vice-versa: using the same name while referring to different notions (*polysemy*). This unfortunately happens too often: see for example the response received on MathOverflow when I asked the community about instances of this phenomenon in their field of mathematical research [44]. This is obviously one of the sources of errors for the mathematical literature that we hope to eliminate thanks to formalisation, nevertheless it can cause problems for the actual formalisation process as well because of the challenges it creates for the organisation of the Library.

(3) Efficient search for proof patterns and algorithms is another big challenge to be tackled. To this end, employing tools from machine learning is considered very promising.

Within our project, Yiannos Stathopoulos and I are currently working on SErAPIS (: **S**earch **E**ngine by the **A**LEXANDRIA **P**roject for **Is**abelle), a concept-oriented search engine [59, 60], by using techniques from natural language processing. A different search engine for Isabelle is independently being developed by Fabian Huch and Alexander Krauss in Munich [20].

4.3 Automation

The high level of detail that is required when formalising a mathematical proof can render the process time-consuming, thus efficient automation is vital. This is one important factor that makes Isabelle/HOL more user-friendly than other proof assistants. In general, Sledgehammer and `try0` work remarkably well for very simple statements, while the possibility of counterexample finding with Quickcheck and nitpick can be very helpful. However, in many occasions certain elementary examples (e.g. splitting up summations) cannot be solved by automation as one would initially expect. Another observation is that for algebraic expressions where minus is involved, it is often much easier to find proofs by automation when we are working with type integer than when we are working with type natural, due to the respective definitions not having the same algebraic properties. Moreover, in certain cases where long and complicated expressions are involved and in examples involving distributivity, simplification, handling inequalities involving division/multiplicative inverses, comparing inequalities after multiplying with a complicated expression, comparing compli-

cated expressions involving logarithms or exponents, Sledgehammer times out (while `try0` also fails), even though in examples of the same style but involving simpler expressions they are successful in finding proofs. It is worth noting that automation power is constantly getting improved, so in a couple of years, if not sooner, more and more complicated examples will most likely be solvable by automation.

As a sidenote regarding automation, a common source of errors for inexperienced users is the use of the keyword `sorry` which temporarily regards an unproven statement as proven, so that the user can go on with the formalisation and even use the statement in other proofs, and return later to fill in the gap. This may lead automation to suggest a proof of a statement B using a wrong statement A that had been "shown" by `sorry` prior to statement B. This is a consequence of the principle of explosion in logic (ex falso sequitur quodlibet–from falsehood anything follows). Even though `sorry` is extremely helpful, new users should be aware of the above issue.

4.4 Using the Formalised Material

Finally, other issues of secondary importance that the user should be aware of relate to the rigidity of the structure of the formalised fundamental material in the Library. For instance, the definition of an "algebra" in the Library includes associativity, so working with nonassociative algebras is not possible within an existing type class, e.g. see my development of Octonions [34] following Lawrence Paulson's development of Quaternions [52]. Another example is an observation by Anthony Bordg that a ring-scheme is required in order to define an abelian monoid.[10]

It should be noted that in my experience so far I did not encounter a problem with the fact that Isabelle is based on simple types instead of dependent types. Lawrence Paulson discusses this issue [53].

5 Experimenting and Exploring

For the first project I suggested to formalise a proof that is (1) research-level, (2) using elementary material that is already in the Isabelle Library, (3) not very long or too complicated. I thus opted for an irrationality criterion for infinite series by Hančl [18] that fulfills the above prerequisites and Wenda Li and I formalised it [38]. It turned out that the Library did not contain a development for infinite products that were used in the proof (at that time we were using Isabelle 2017), so we used infinite sums via the logarithm instead. With this incentive, Lawrence Paulson soon afterwards wrote a development for infinite products (extending earlier work by Manuel Eberl) which was included in Isabelle 2018.

After a discussion with Anthony Bordg I realised that it can also be meaningful to formalise a theorem even if all the proofs of the statements its proof relies on are yet not fully formalised. This would be achieved by using the theorems on which our theorem at hand depends as assumptions (implemented as a *locale*, or within a *context*).

[10] Isabelle mailing list archive, available on https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2018-November/msg00052.html.

This is a compromise on the vision of "absolute correctness" but it is a more realistic approach that reflects actual mathematical practice, and at least provides verification on a level of "relative" correctness. An obvious benefit of this approach would be the possibility to reach more advanced mathematics more quickly. I am still too reluctant to recommend this, as a generalised use of it could help spread errors in the literature undermining the whole point of verification and even do more harm than good if the wrong results have the "Isabelle/HOL-verified" seal. But provided that (1) the theorem used as an assumption is safe as a central/mainstream result that has been widely and carefully checked and (2) assumptions and methods are made fully transparent, it can be a worthwhile experiment. I thus decided to formalise some result in research-level mathematics following from some well-known, fundamental theorem the proof of which has not been formalised, so Wenda Li and I formalised two transcendence criteria for infinite series by Hančl and Rucki [19] that make use of Roth's celebrated theorem on rational approximations [55]. In our formalisation [39], Roth's theorem was given only as an assumption. The process of formalisation revealed a slight mistake in the original proof [19] which, as it turned out after corresponding with the authors, was very easily fixable. Wenda Li and I have also recently formalised certain irrationality criteria for infinite series by Erdős and Straus from 1974 [10]. As noted in our formalisation work [40], certain inequalities from one of the proofs of the original paper required a few minor corrections (this did not affect the correctness of the statements nor the general proof structure).

While formalising Aristotle's assertoric syllogistic in Isabelle/HOL [33] (following [57]), I noticed that Sledgehammer was very efficient: it is interesting that it could find straightforward one-line proofs for statements that ancient philosophers showed with much lengthier arguments. In mathematics it is, of course, usually the opposite, formal proofs are longer than the original; the de Bruijn factor is used to measure how much longer a formal proof is, and it is defined as the ratio of the number of lines of the formal proof to the number of lines of the corresponding informal proof in TeX encoding [63, 64].

I have also recently formalised some material on amicable numbers [35], which revealed the efficiency of certain methods with respect to numerical computations. In particular, the method `simp` immediately shows primality for small prime numbers (for larger primes, to show primality I make use of Pratt Certificate [67] which is also straightforward to apply) and the method `eval` helps to show what is the set of all divisors of a number.

Another direction that currently interests me is a possible use of formalisation for the benefit of automating pen-and-paper proof mining, which I conjecture could be achieved once machine learning comes into play. I suggest, in particular, that while building extensive libraries of formal mathematical proofs, it would be meaningful to opt for formalising proofs whose computational content is made explicit in the meantime, so that as automation improves and blocks of (sub)proofs get generated automatically, the preserved computational content would get recycled, recombined and would eventually manifest itself in different contexts. To this end, I do not suggest restricting to only constructive proofs, but I suggest that proof mined (i.e. possibly also non-constructive proofs but with some explicit computational content) should be preferable, if possible [36, 37].

6 Disclaimers and Cautions

Using a proof assistant to verify mathematics is somewhat reminiscent of writing down a relative consistency proof as logicians do. (Of course, this is not a problem for formalists.[11] Timothy Gowers claims that "modern mathematicians are formalists, even if they profess otherwise" and that "it is good that they are" [15]. Kevin Buzzard, as many mathematicians interested in automated reasoning, also describes himself as a formalist).

This reminiscence can be seen on two different levels: (1) From the point of view of the correctness of the core of the system. This can be trusted considering the underlying architecture of the proof system and the small size of the core [51, 53] and as (on a more practical level) the user interface provides a window showing the proof state at every step so the user can be in control of the proof. (2) From the point of view of the correctness of the mathematical assumptions, which is the issue I discuss below. Using a proof assistant does not guarantee "absolute" correctness, as the work might always be prone to human errors. An obvious way of making a bad use of a proof assistant thus leading to human errors going undetected is proving something different than what was initially intended or claimed (while the intended statement might even be false) because of either (a) using a misleading name of the proved statement/ misleading definitions [44] or (b) something as simple as a typographical error in an intermediate step of the proof like a misplaced parenthesis e.g. showing $f(n+1)$ instead of $f(n)+1$. Also mind the principle of explosion in logic (from falsehood anything follows)!

6.1 Different Kinds of "Wrong" in Mathematics and Isabelle/HOL

This part is to be read as a caution to new proof assistant users; as already explained, using a proof assistant is not a panacea guaranteeing correctness, as the work might always be prone to human errors which, as we illustrate in the examples below, may not be detected by the proof assistant. Mathematicians may make mistakes that fall into different categories. Several (extremely naive) examples for each category of mistakes and "mistakes" that may be committed by human mathematicians are given below, and it is shown how Isabelle may not necessarily detect them, as they are essentially "semantic" mistakes that are syntactically correct. Of course, a mathematician would not make the specific elementary mistakes mentioned below, but mistakes of a similar nature in a much more sophisticated context could possibly be made, and as is the case with the naive mistakes below, they may be similarly overlooked.

[11] Here we refer to formalism as the philosophical view that mathematical statements can be considered as statements about the consequences of the manipulation of symbols using established rules and that ontologically mathematics is much more akin to a game than a representation of some objective reality.

6.1.1 Proving a Conclusion That Is Too General

```
theorem wrongroot:
  fixes x::real
  assumes "x^2 -1=0"
  shows "x=1∨x=-1∨x=4"
  using assms
  apply (auto simp add:power2_eq_square algebra_simps)
  using square_eq_1_iff by blast
```

This is logically correct, but mathematically undesirable. This is accepted by Isabelle as an instance of the logical axiom $A \longrightarrow A \vee B$.

6.1.2 Using a Superfluous Assumption

```
theorem superfluous:
  fixes x y z a::real
  assumes "x= (y-1)^2 +4*z" and "y=a+2" and "z=5*y"
  shows "x= (y-1)^2 +20*y"
  using assms apply simp done
```

This is again logically correct, but mathematically undesirable and accepted by Isabelle as an instance of $A \wedge B \longrightarrow B$ from logic. (As a sidenote, the proof method `metis` in particular does warn the user about unused theorems in the proof. Moreover, users can check themselves for unused assumptions with `find_unused_assms`, but this is not always efficient).

6.1.3 A Logical Inconsistency in the Assumptions

We have seen that superfluous assumptions are permitted–moreover it can be that some might even contradict each other:

```
theorem incoherent:
  assumes "∀x. P x" and "∃x. ¬P x"
  shows "∀x. P x"
  using assms by auto
```

An instance of the above in mathematics, for example would be:

```
theorem incoherentexample:
  fixes z y::nat and f::"nat ⇒ nat"
  assumes "∀k. ∃x.(f x <2^k)" and "z<y"
   and "∃k. ∀x.(f x ≥ 2^k)"
  shows "∀k. ∃x.(f x < 2^k)" and "z<2*y"
  using assms by auto
```

6.1.4 *Assuming One (or More) Wrong Fact(s)/ Assumption(s) That Cannot Be Fulfilled*

```
theorem wrongfact:
  fixes y z::nat
  assumes "prime(y*z)" and "y>2" and "z>2"
  shows "prime(y*z*1)"
  using assms by auto

theorem wrongfact1:
  assumes "pi ∈ Rats" and "k ∈ Rats"
  shows "pi*k ∈ Rats"
  using assms by auto

theorem wrongfact2:
  assumes "∀ x y ::nat. prime(x) ∧ prime(y)⟶prime(x-y)"
  shows "prime((8::nat))"
  using assms(1)[rule_format, of 13, of 5 ] by auto

theorem unfulfilledassum:
  fixes  f::"nat⇒nat"
  assumes "∀ y x.(x ≤ y⟶f y ≤ f x)"
    and "f = (λx. 1+x)"
  shows "f 2 ≤ f 1"
  using assms by auto
```

The first examples are instances of assuming a wrong mathematical fact while the last one is an instance of the assumptions being incompatible as in the above subsection. For example, such mistakes (unless of course we are using assumptions that we conjecture to be false so that they yield a proof by contradiction) could occur when an author makes bad use of the literature by using two or more nonequivalent definitions of a mathematical object within the same context, which may easily happen [44].

6.1.5 *Lack of Precision when Approximating*

This instance of course falls into the aforementioned case of assuming wrong facts, but this demonstrates how imprecise approximations (which a user may be often tempted to make, especially when dealing with numerical computations) in particular can be accepted by Isabelle without any error messages, potentially leading to further wrong numerical results.

```
theorem notprecise1:
  fixes y::"real⇒real"
  assumes "y = (λx. cos x + sin x)"
  and "∀ x. (sin x=x-x^3/6 ∧ cos x=(1::real)-x^2/2)"
  shows "y x = x+1-x^3/6-x^2/2"
  using assms by auto

theorem badapproxi:
  assumes "∀ (x::real). (sin x=x)"
  shows "sin (10::real)=(10::real)"
  using assms by auto
```

6.1.6 Requirement of an Additional Assumption

Mathematical practice involves trial-and-error (Lakatos [41]). A human mathematician would try to figure out if they could prove their conjecture to be true after adding or removing assumptions. Although often Isabelle helps towards this direction by finding counterexamples thanks to nitpick or Auto Quickcheck, usually the user would not directly receive a suggestion by Isabelle to modify the assumptions. For example, in the following it would be useful if Isabelle would explicitly suggest to include the assumption $d \neq 0$ (if this is included, a proof by simp is immediately found):

```
theorem simply:
  fixes a b c d::real
  assumes "a * b = c * d"
 (* and "d ≠0"*)
  shows " c = (a * b)/d" sorry
 (* using assms apply simp done*)
```

6.2 Discussion

Examples 6.1.1-6.1.4 should not be considered surprising: we can view them as a manifestation of the fact that Isabelle relies on logical inference [51] while mathematics, even though it is "a method of logic" as Wittgenstein wrote [68], is, moreover, richer in content and meaning. From a logical point of view $A \wedge B \rightarrow B$ is fundamental, but from a mathematical point of view assuming $A \wedge B$ (that could internally amount to B if assumption B would be sufficient to prove some specific end goal) would appear as making use of a superfluous assumption which is undesirable; from a logical point of view the principle of explosion (from falsehood anything follows) is fundamental, but in mathematics proving a false conclusion from assumptions that we already know are false would be a senseless pastime (except, of course, in the case of assumptions that are only conjectured to be false when writing a proof by contradiction); from a logical point of view $A \rightarrow A \vee B$ is a fundamental axiom, but from a mathematical point of view it would not be an interesting deduction to make.

Examples 6.1.5 and 6.1.6 are not surprising either, as for Isabelle to suggest a modification of the assumptions as a human mathematician would do would require Isabelle to feature a kind of "Intelligence" that no proof assistant nowadays features (not yet?). Russell and Norvig [56] (Chap. 1.1) distinguish four different categories of definitions of artificial intelligence, none of which is safe to say that is currently fully satisfied by Isabelle, nor by any other proof assistant to my knowledge.

The conclusion to draw from the above examples is that, after all, the user is the one in control and thus should be careful to use Isabelle (or any other proof assistant) responsibly. It should be stressed that the high value of a proof assistant lies in the fact that it prevents another very common and dangerous source of fallacy in mathematical practice: *being misled by one's intuition* thus allowing for gaps in a proof or for overlooking mistakes. A proof assistant like Isabelle would *not* allow for this to happen, as every argument is required to be made explicit in all detail. In my

own experience working in proof mining, which is pen-and-paper work revealing the hidden logical structure of proofs, it sometimes turns out that the original proof can be written in a more detailed and structured way. However, working in automated reasoning reveals the requirement of an even higher level of rigour. Another benefit follows from the flexibility and interactive nature of working with code in general: one can more easily uncover hidden dependencies, e.g. one can see how and where a proof breaks by making changes such as deleting assumptions.

7 Towards the Future

Is the way that we are doing mathematics being revolutionised? Most likely, yes.

This is reflected in an important recent milestone; for the first time, a class titled "Computer science support for mathematical research and practice" is included in the new Mathematics Subject Classification announced in early 2020 (MSC 2020, Class 68Vxx). This class includes topics such as computer assisted proofs, proofs employing automated/ interactive theorem provers, formalisation of mathematics in connection with theorem provers. Thus, the aforementioned topics are now officially recognised as areas of mathematical research (in addition to computer science research as it was until now).

As an anecdote indicative of the current climate I mention that during the panel discussion of the workshop *Foundations in Mathematics: Modern Views* in April 2018 in Munich[12] that attracted young (mostly student-level) mathematicians, philosophers and logicians, interestingly the dominant view discussed arguing for the importance of exploring the foundations of mathematics was their significance for computerised mathematical proofs which among the participants of the discussion was regarded as an inevitable development.

This revolution in mathematical practice can be regarded as two-fold: The first aspect is associated with our expectation of a certain level of correctness and thus refers more to the benefits of formalisation for the purpose of verification. Barendregt and Wiedijk [4] beautifully summarise this vision in the citation given in the beginning of this article.

The second aspect of the revolution is associated with the tools used in our day-to-day work. A groundbreaking tool would be an interactive assistant that would provide working mathematicians with brainstorming in the form of tips on how to prove a statement, or even suggestions for new conjectures. This would be achieved by implementing machine learning tools in an appropriate way, and after a substantial amount of mathematics has been formalised. Today we are far from this goal, but even before we approach it, just typing mathematics in Isar (or any other machine language) instead of using paper and pen is already a big shift in our way of working, which would inevitably influence our way of thinking. We are one of the first generations that now type much more than using a pen in our daily lives, and it is conceivable that our strong impulse to write with a pen or a piece of chalk while shaping a brand new thought or trying to understand an idea may disappear in future generations. This

[12] https://fmv2018.weebly.com/.

is of course an issue related to psychology and cognitive science rather than purely to mathematical practice. As a striking example it should at least be noted that most mathematicians nowadays have no trouble understanding LaTeX code, while a few years back it was considered rather unusual, and most people would agree that Isar looks not more unnatural than LaTeX.[13]

Aristotle in his Nicomachean Ethics wrote [2] (Book VII, Chap. 8, 17-19):

"In mathematics, it is not reason that teaches first principles, nor is it in actions; rather, it is virtue, either natural or habituated, that enables us to think correctly about the first principle."

Could it be that, one day, current mathematical practice might seem to the future generations as antiquated (or as peculiar), as the above statement may sound to us nowadays?

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK. Thanks to Lawrence Paulson for reading and commenting on an earlier version of this paper, to the anonymous reviewers for their useful suggestions and to Lawrence Paulson, Wenda Li, Manuel Eberl, Jeremy Avigad, Kevin Buzzard, Peter Koepke, Anthony Bordg, Edward Ayers and Mohammad Abdulaziz for many interesting discussions. The workshop *Computer Aided Mathematical Proof*[14] at the Isaac Newton Institute for Mathematical Sciences in Cambridge, UK, organised by Ursula Martin, Lawrence Paulson and Andrew Pitts, that I attended in July 2017, as well as a talk by Gyesik Lee at *Colloquium Logicum* in Hamburg (September 2016)[15] sparked my early interest in formal verification.

References

1. Archive of Formal Proofs: https://www.isa-afp.org. Accessed 2 September 2020
2. Aristotle: Nicomachean Ethics. In: Crisp, R. (Ed.): Cambridge University Press, Cambridge (2014). Revised edition
3. Avigad, J.: The mechanization of mathematics. Not. Am. Math. Soc. **65**(6), 681–690 (2018). https://doi.org/10.1090/noti1688
4. Barendregt, H., Wiedijk, F.: The challenge of computer mathematics. Philos. Trans. - Royal Soc., Math. Phys. Eng. Sci. **363**(1835), 2351–2375 (2005)

[13] Benedikt Löwe gave an analysis of the evolution of the use of LaTeX by mathematicians over the years and made a comparison with proof assistants during his talk at *Big Proof* which took place in May 2019 at the International Center for Mathematical Sciences in Edinburgh https://www.icms.org.uk/bigproof.php.

[14] https://www.newton.ac.uk/event/bprw01/.

[15] https://www.math.uni-hamburg.de/spag/ml/CL2016/.

5. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, C.: Hammer-ing towards QED. J. Formaliz. Reason. **9**(1), 101–148 (2016)
6. Boyer, R., et al.: The QED manifesto. In: Bundy, A. (Ed.) Automated Deduction – CADE 12. LNAI, vol. 814, pp. 238–251. Springer, Berlin (1994)
7. Buzzard, K.: The Xena Project: Mathematicians learning Lean by doing (blog). https://xenaproject.wordpress.com. Accessed on 4 September 2020
8. Buzzard, K.: Computers and mathematics. Newsletter of the London Mathematical Society, Issue 484 (2019)
9. Buzzard, K., Commelin, J., Massot, P.: Formalising perfectoid spaces (2019). Submitted preprint, arXiv:1910.12320
10. Erdős, P., Straus, E.G.: On the irrationality of certain series. Pac. J. Math. **55**(1), 85–92 (1974)
11. Freiberger, M.: Pure Maths in Crisis? +plus magazine. https://plus.maths.org/content/pure-maths-crisis (2019). Submitted on 19 March 2019, accessed on 4 Sept. 2020
12. Gonthier, G.: Formal proof–the Four-Color Theorem. Not. Am. Math. Soc. **55**(11), 1382–1393 (2008)
13. Gouëzel, S., Shchur, V.: A corrected quantitative version of the Morse lemma. J. Funct. Anal. **277**(4), 1258–1268 (2019)
14. Gowers, T.: Rough structure and classification. In: Alon, N., Bourgain, J., Connes, A., Gromov, M., Milman, V. (Eds.) Modern Birkhäuser Classics. Visions in Mathematics Birkhäuser, Basel (2010)
15. Gowers, T.: Does Mathematics Need a Philosophy? https://www.dpmms.cam.ac.uk/~wtg10/philosophy.html. Accessed 4 September 2020
16. Hales, T.: A proof of the Kepler conjecture. Ann. Math. **162**(3), 1065–1185 (2005)
17. Hales, T., Adams, M., Bauer, G., Dang, T.D., Harrison, J., Hoang, L.T., Kaliszyk, C., Magron, V., McLaughlin, S., Nguyen, T.T., Nguyen, Q.T., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., Ta, T.H.A., Tran, N.T., Trieu, T.D., Urban, J., Vu, K., Zumkeller, R.: A formal proof of the Kepler conjecture. In: Forum of Mathematics, Pi, vol. 5, pp. e2 (2017)
18. Hančl, J.: Irrational Rapidly Convergent Series. Rend. Sem. Mat. Univ. Padova, vol. 107 (2002)
19. Hančl, J., Rucki, P.: The transcendence of certain infinite series. Rocky Mt. J. Math. **35**(2), 531–537 (2005)
20. Huch, F., Krauss, A.: FindFacts: a scalable theorem search. In: Online proceedings of the Isabelle Workshop 2020, affiliated to IJCAR 2020 (in Virtual Space), June 30, 2020. https://files.sketis.net/Isabelle_Workshop_2020/Isabelle_2020_paper_3.pdf
21. Isabelle Website: https://www.cl.cam.ac.uk/research/hvg/Isabelle/index.html. Accessed 4 September 2020
22. Isabelle/HOL Library: http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/library/HOL/index.html. Accessed 4 September 2020
23. Isabelle/ZF Library: https://isabelle.in.tum.de/dist/library/ZF/index.html. Accessed 4 September 2020
24. Koepke, P.: Mathematics and the new technologies, part II: computer-assisted formal mathematics and mathematical practice. In: Schröder-Heister, P., Heinzmann, G., Hodges, W., Bour, P.E. (Eds.) Logic, Methodology and Philosophy of Science - Proceedings of the 14th International Congress (2014)
25. Kohlenbach, U.: Applied Proof Theory: Proof Interpretations and Their Use in Mathematics. Springer Monographs in Mathematics. Springer, Berlin (2008)
26. Kohlenbach, U.: Recent progress in proof mining in nonlinear analysis. IfCoLog J. Log. Appl. **10**(4), 3361–3410 (2017)
27. Kohlenbach, U.: Proof-theoretic methods in nonlinear analysis. In: Sirakov, B., Ney de Souza, P., Viana, M. (Eds.) Proceedings of the International Congress of Mathematicians 2018, vol. 2, pp. 61–82. World Scientific, Singapore (2019)
28. Kohlenbach, U., Koutsoukou-Argyraki, A.: Rates of convergence and metastability for abstract Cauchy problems generated by accretive operators. J. Math. Anal. Appl. **423**, 1089–1112 (2015)
29. Kohlenbach, U., Koutsoukou-Argyraki, A.: Effective asymptotic regularity for one-parameter nonexpansive semigroups. J. Math. Anal. Appl. **433**, 1883–1903 (2016)
30. Koutsoukou-Argyraki, A.: Effective rates of convergence for the resolvents of accretive operators. Numer. Funct. Anal. Optim. **38**(12), 1601–1613 (2017)
31. Koutsoukou-Argyraki, A.: New effective bounds for the approximate common fixed points and asymptotic regularity of nonexpansive semigroups. J. Log. Anal. **10:7**, 1–30 (2018)
32. Koutsoukou-Argyraki, A.: Proof Mining for Nonlinear Operator Theory: Four Case Studies on Accretive Operators, the Cauchy Problem and Nonexpansive Semigroups. PhD thesis, TU Darmstadt (2017). URN:urn:nbn:de:tuda-tuprints-61015. https://tuprints.ulb.tu-darmstadt.de/id/eprint/6101

33. Koutsoukou-Argyraki, A.: Aristotle's Assertoric Syllogistic. Archive of Formal Proofs (2019). Formal proof development. https://www.isa-afp.org/entries/Aristotles_Assertoric_Syllogistic.html
34. Koutsoukou-Argyraki, A.: Octonions. Archive of Formal Proofs (2018). Formal proof development. https://www.isa-afp.org/entries/Octonions.html
35. Koutsoukou-Argyraki, A.: Amicable Numbers. Archive of Formal Proofs (2020). Formal proof development. https://www.isa-afp.org/entries/Amicable_Numbers.html
36. Koutsoukou-Argyraki, A.: Proof mining mathematics, formalising mathematics. Bull. Symb. Log. **24**(4) (2018). Proceedings of the North American Annual Meeting of the Association for Symbolic Logic, University of Western Illinois, Macomb, Illinois, USA, May 16-19, 2018
37. Koutsoukou-Argyraki, A.: On preserving the computational content of mathematical proofs: toy examples for a formalising strategy. In preparation
38. Koutsoukou-Argyraki, A., Li, W.: Irrational rapidly convergent series. Archive of Formal Proofs (2018). Formal proof development. https://www.isa-afp.org/entries/Irrationality_J_Hancl.html
39. Koutsoukou-Argyraki, A., Li, W.: The transcendence of certain infinite series. Archive of Formal Proofs (2019). Formal proof development. https://www.isa-afp.org/entries/Transcendence_Series_Hancl_Rucki.html
40. Koutsoukou-Argyraki, A., Li, W.: Irrationality Criteria for Series by Erdős and Straus. Archive of Formal Proofs (2020). Formal proof development. https://www.isa-afp.org/entries/Irrational_Series_Erdos_Straus.html
41. Lakatos, I.: In: Proofs and Refutations, 1st edn. Worrall, J., Zahar, E. (Eds.): Cambridge University Press, Cambridge (1976)
42. Lamport, L.: How to write a 21st century proof. J. Fixed Point Theory Appl. **11**, 43 (2012)
43. Löwe, B.: Mathematics and the new technologies, part I: philosophical relevance of a changing culture of mathematics. In: Schröder-Heister, P., Heinzmann, G., Hodges, W., Bour, P.E. (Eds.) Logic, Methodology and Philosophy of Science - Proceedings of the 14th International Congress (2014)
44. The MathOverflow Community: Nonequivalent definitions in Mathematics, Question asked by Koutsoukou-Argyraki, A. on 22 Nov. 2017. https://mathoverflow.net/questions/286732/nonequivalent-definitions-in-mathematics
45. Negri, S., Plato, J.V.: Proof Analysis–A Contribution to Hilbert's Last Problem. Cambridge University Press, Cambridge (2011)
46. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science, vol. 2283. Springer, Berlin (2002)
47. Paulson, L.C.: ALEXANDRIA: Large-scale formal proof for the working mathematician. Project description (12 Nov. 2018). http://www.cl.cam.ac.uk/~lp15/Grants/Alexandria/
48. Paulson, L.C.: A machine-assisted proof of Gödel's incompleteness theorems for the theory of hereditarily finite sets. Rev. Symb. Log. **7**(3), 484–498 (2013)
49. Paulson, L.C.: A mechanised proof of Gödel's incompleteness theorems using nominal Isabelle. J. Autom. Reason. **55**, 1–37 (2015)
50. Paulson, L.C.: Computational logic: its origins and applications. Proc. R. Soc. A, Math. Phys. Eng. Sci. **474**(2210), 20170872 (2018)
51. Paulson, L.C.: The foundation of a generic theorem prover. J. Autom. Reason. **5**(3), 363–397 (1989). Kluwer Academic Publishers
52. Paulson, L.C.: Quaternions. Archive of Formal Proofs (2018). Formal proof development. https://www.isa-afp.org/entries/Quaternions.html
53. Paulson, L.C.: Formalising mathematics in simple type theory. In: Sarikaya, D., Kant, D., Centrone, S. (Eds.) Reflections on the Foundations of Mathematics. Springer, Berlin (2019)
54. Pöppe, C.: Hilbert und Isabelle. Spektrum.de. 20.02.2019 (in German). https://www.spektrum.de/magazin/optimierung-diophantischer-gleichungen/1621174?utm_medium=newsletter&utm_source=sdw-nl&utm_campaign=sdw-nl-mag
55. Roth, K.F.: Rational approximations to algebraic numbers. Mathematica **2**(3), 1–20 (1955)
56. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. (2010). Pearson Publishing
57. Smith, R.: Aristotle's Logic. The Stanford Encyclopedia of Philosophy (website). First published 18 March 2000. Substantive revision 17 February 2017. https://plato.stanford.edu/entries/aristotle-logic/
58. Stock, B., et al.: Hilbert Meets Isabelle: Formalisation of the DPRM Theorem in Isabelle. EasyChair Preprint No. 152, EasyChair (2018). https://doi.org/10.29007/3q4s
59. Stathopoulos, Y., Koutsoukou-Argyraki, A., Paulson, L.C.: SErAPIS: A Concept-Oriented Search Engine for the Isabelle Libraries Based on Natural Language. In: Online proceedings of

the Isabelle Workshop 2020 affiliated to IJCAR 2020 (in Virtual Space), June 30, 2020. https://files.sketis.net/Isabelle_Workshop_2020/Isabelle_2020_paper_4.pdf
60. Stathopoulos, Y., Koutsoukou-Argyraki, A., Paulson, L.C.: Developing a Concept-Oriented Search Engine for Isabelle Based on Natural Language: Technical Challenges. In: Online proceedings of the 5th Conference on Artificial Intelligence and Theorem Proving (AITP 2020), Aussois, France, March 22-27 (event postponed to September 13-18, 2020). http://aitp-conference.org/2020/abstract/paper_9.pdf
61. Voevodsky, V.: The Origins and Motivations of Univalent Foundations. The Institute Letter, Institute of Advanced Studies (2014). https://www.ias.edu/sites/default/files/pdfs/publications/letter-2014-summer.pdf
62. Whitehead, A.N., Russell, B.: Principia Mathematica, vols. 1, 2, 3. Cambridge University Press, Cambridge (1927)
63. Wiedijk, F.: The De Bruijn Factor (2000). http://www.cs.ru.nl/F.Wiedijk/factor/factor.pdf
64. Wiedijk, F.: The De Bruijn Factor (Webpage). http://www.cs.ru.nl/~freek/factor/. Accessed on 4 September 2020
65. Wiedijk, F.: The Seventeen Provers of the World, forward by Dana S. Scott. Lecture Notes in Artificial Intelligence, vol. 3600. Springer, Berlin (2006)
66. Wiedijk, F.: Formalizing 100 Theorems (Webpage). http://www.cs.ru.nl/~freek/100/. Accessed 4 September 2020
67. Wimmer, S., Noschinski, L.: Pratt's Primality Certificates. Archive of Formal Proofs (2013). Formal proof development. https://www.isa-afp.org/entries/Pratt_Certificate.html
68. Wittgenstein, L.: Tractatus Logico-Philosophicus (Logisch-Philosophische Abhandlung). In: W. Ostwald's Annalen der Naturphilosophie, 1st edn. (1921)

Angeliki Koutsoukou-Argyraki received her PhD in pure mathematics from Fachbereich Mathematik, Technische Universität Darmstadt, Germany, in 2016. She was a postdoctoral scientific associate at the Logic Group there until 2017. Her research involves applications of logic to various areas of mathematics (in particular proof mining, a programme in applied proof theory involving pen-and-paper extraction of computable information from mathematical proofs). Since 2017 she is a postdoctoral research associate at the Department of Computer Science and Technology (Computer Laboratory), University of Cambridge, UK, where she works on formalisation of mathematics with proof assistants/interactive theorem proving, within the Automated Reasoning Group.