Synthese manuscript No.
(will be inserted by the editor)

Computer Verification for Historians of Philosophy

Landon D. C. Elkind

March 18, 2022

Abstract Interactive theorem provers might seem particularly impractical in the history of philosophy. Journal articles in this discipline are generally not formalized. Interactive theorem provers involve a learning curve for which the payoffs might seem minimal. In this article I argue that interactive theorem provers have already demonstrated their potential as a useful tool for historians of philosophy; I do this by highlighting examples of work where this has already been done. Further, I argue that interactive theorem provers can continue to be useful tools for historians of philosophy in the future; this claim is defended through a more conceptual analysis of what historians of philosophy do that identifies argument reconstruction as a core activity of such practitioners. It is then shown that interactive theorem provers can assist in this core practice by a description of what interactive theorem provers are and can do. If this is right, then computer verification for historians of philosophy is in the offing.

Keywords Formal methods · Formalization · History of philosophy · Interactive theorem provers · Metaphilosophy

1 Introduction

In this paper I discuss the place of some computational formal methods in doing history of philosophy. Specifically, I describe how to apply interactive theorem provers—these are (software) applications of a certain sort described more fully in §2—in textual interpretation and argument reconstruction, to the benefit of both researchers and their broader scholarly community.[1] More

Landon D. C. Elkind
Department of Political Science, Cherry Hall 306; Western Kentucky University; 1906 College Heights Blvd; Bowling Green, KY 42101-1086; USA
E-mail: landon.elkind@wku.edu
ORCID 0000-0003-0513-2937

[1] In this paper I will use "interactive theorem provers" and "proof assistants" as rough synonyms, although there are non-interactive theorem provers; see §2.

concretely, such applications involve formalizing key notions in the argument or text in a manner that the file can read and understood, compiling the file and fixing any runtime errors, and then identifying the philosophical results of executing a file in the interactive theorem prover. This last step usually involves producing a human-readable writeup of what was done, problems found, solutions implemented, and lessons learned, perhaps with some excerpted code from the application file.

All of this can occur alongside and as a supplement to research produced using more traditional, informal, or non-computational methods in history of philosophy, that is, interactive theorem provers are not a replacement or substitute for critically thinking about a text as historians of philosophy have done for thousands of years, nor are they a replacement for longstanding methods in history of philosophy. Rather, deploying interactive theorem provers can complement and support historians' usual activities, especially by automating for readers much of the mental labor of verifying arguments and spotting informal and formal fallacies. Perhaps most importantly, utilizing interactive theorem provers can spare others the labor of rewriting formalized arguments again once it has been done once because the formalization sources from argument reconstructions in interactive theorem provers can be open-source. Thus, such code can be downloaded, modified, retooled, and fit to new purposes.

For any historian of philosophy, particularly if one is unfamiliar with interactive theorem provers, the natural question to ask at this stage is, 'Why would I do all of that?' What I described sounds like much more work for not terribly much payoff. After all, arguments of past philosophers can be reconstructed and even formalized on paper, as they have been for over a century, without the need to translate them into some interactive theorem prover's system. Such translations might even negatively effect the work: whichever formal system is used within some interactive theorem prover may have a distorting effect on the past philosopher's argument. So such applications of interactive proof assistants appear at first blush to involve much work with little or no gain, and, as a historian of philosophy sensitive to issues of translation is well-positioned to notice, could even be a substantial step backwards.

In this paper I address these concerns. My view is that interactive theorem provers have already been, and stand to continue being, useful to historians of philosophy. So much may seem obvious after reviewing some research in history of philosophy that leverages interactive theorem provers, which I do below. The novelty in my argument here is to indicate the untapped potential of interactive theorem provers to historians of philosophy. Interactive theorem provers cannot do philosophy for us, or, to make a more modest claim, nothing in my argument hinges on claiming that they can. But the ways in which interactive theorem provers can assist research in history of philosopher are about as plenitudinous as the ways in which computer verification and sharing code assists software development. That is what I argue for here. If this conclusion is true, then interactive theorem provers can be very useful tools indeed.

This paper builds on work by other philosophers in a similar vein, especially those applying interactive theorem provers in philosophy. There have been at

multiple applications of the contemporary metaphysical and epistemological notions in philosophy. For example: Fitelson and Zalta (2007) have done axiomatic metaphysics in the interactive theorem prover *Prover9*. Benzmüller et al. (2015) have formalized various modal systems and the relations between them in the interactive theorem prover *Isabelle/HOL*. Novak (2015) has used the computer proof-assistant MetaPRL to formalize certain epistemological notions and then used that formalization in MetaPRL to analyze well-known puzzles like the Surprise Examination Paradox. Blumson (2021) has axiomatized classical mereology in *Isabelle/HOL*.

Additionally, interactive theorem provers have been applied to the texts of past figures, including philosophers. For example: Fleuriot (2001) has formalized arguments in Isaac Newton's *Principia Mathematica* using interactive theorem provers. Lokhorst (2011) has formalized Mally's deontic logic and meta-ethical principles in the interactive theorem prover *Prover9*. Alama et al. (2015) have formalized (an interpretation of) Leibniz's theory of concepts in the *E* and *Vampire* theorem provers (plus the *Paradox* model searching program). Benzmüller and Paleo (2015) and Fuenmayor and Benzmüller (2017) have formalized multiple readings of Gödel's ontological argument for the existence of God in *Isabelle/HOL*. Building on the informal work in (Smith 2020), Koutsoukou-Argyraki (2019) has formalized in *Isabelle* some of Aristotle's proofs and meta-theoretical results concerning his syllogistic.

Citing all these developments, Kirchner et al. (2019, §4) have defended the "benefit from interdisciplinary studies in which computational techniques are applied" and shown some use for interactive theorem provers in metaphysics. Fuenmayor and Benzmüller (2018a) discuss the use of interactive theorem provers in formalizing natural language arguments and describe their approach as "computational hemeneutics." As yet, though, philosophers have not considered the general applicability of interactive theorem provers in doing history of philosophy, especially by reference to the scholarly activities of historians of philosophy and to the specific issues raised in applying formal methods, including computational ones like interactive theorem provers, in doing history of philosophy. This lacunae exists in the literature despite the fact that answers to some significant methodological issues are implicitly assumed in some applications of interactive theorem provers just noted, especially in the formalizations of Leibniz's theory of concepts and Gödel's argument for the existence of God. Hence, there is a real need for the present essay.

First I briefly describe what interactive theorem provers are (§2). The purpose of doing that will be to show how these applications can be used in the philosophical historian's practice of formalizing arguments.[2] Those already familiar with interactive theorem provers might skip this section, referring back

[2] So far as I know, the phrase 'philosophical history' was coined by Michael Kremer (2013, 294). Kremer (2013, 298-299) offers this terminology in a way that builds on Bernard William's distinction between two interrelated endeavors, "history of ideas" and "history of philosophy": history of ideas produces something that is history first and philosophy second, the history of philosophy produces something that is philosophy first and history second (Williams 1994, 19). Since I am convinced of Kremer's claim that philosophical history is a valuable and distinctive intellectual enterprise, and since philosophical history is the enter-

to specific details as needed. Next I discuss the metaphilosophical issues raised by formalizing arguments in doing history of philosophy (§3). There I argue that what is commonly called rational reconstruction of arguments can benefit from formalization using interactive proof assistants, and further, that such argument formalization can serve as a helpful complement to the other kinds of investigation undertaken by historians of philosophy.

Then I discuss some examples of applying interactive theorem provers in history of philosophy (§4). Considering these applications will support my claim in §3 that formalizing arguments using interactive theorem provers can benefit the practice of doing history of philosophy. Finally I tie all of this discussion together to offer a prospective view of what interactive theorem provers can assist historians of philosophy in doing (§5). To give away the ending, computationally verifying argument reconstructions using such applications offers definite benefits to philosophers working in history of philosophy. Thus interactive theorem provers can be a useful tool to an important activity, rational reconstruction, in doing history of philosophy.

2 Interactive theorem provers

What is an interactive theorem prover? This question implicitly raises two difficult and longstanding subsidiary questions: (1) What is a theorem? (2) What is a proof? But we do not need to dive into these trenches. Rather, we can explain interactive theorem provers without having to endorse a specific account of theorems and proofs.[3]

In introductory logic textbooks, philosophers usually say that an *argument* is a collection of sentences, one of which is the conclusion, and the rest of which, if any, are the premises offered in support of the conclusion.[4] For purposes of the argument here, we can remain largely agnostic on further thorny issues in the philosophy of logic and mathematics by considering three subclasses of arguments in this sense. First, *deductive arguments* are those arguments wherein it is logically necessary that, if the premises hold, then the conclusion holds.[5] Second, *axiomatic arguments* are those that ultimately depend

prise that is my focus here, I will use 'philosophical history' and 'philosophical historians' interchangeably with 'history of philosophy' and 'historians of philosophy' respectively.

[3] The below paragraph essentially follows the clean presentation in (Portoraro 2019, §1), though I replace Portoraro's talk of problems and solutions with talk of premises and conclusions.

[4] By focusing on arguments here, it is not implied that historians of philosophy are either primarily or only concerned with arguments in this sense. See §3 for discussion.

[5] In contrast, *non-deductive* arguments are those wherein, probably, if the premises hold, then the conclusion holds. There is significant controversy over how to understand an argument's premises making probable its conclusion, and in particular whether this should be couched in terms of an agent's subjective probabilities (the Bayesian view) or instead, taking the states of affairs wherein the premises hold, in a measurement of how many such scenarios are such that the conclusion holds. See (Hawthorne 2021, §2).

only on axioms and primitive rules for making inferences from these.[6] Third, *formal arguments* are those couched within a *formal language*, that is, "a recursively defined set of strings on a fixed alphabet." (Shapiro and Kissel 2018, §2) Putting all three subclasses together, *interactive theorem provers* are software applications whose inputs are attempts at deductive, axiomatic, formal arguments, and whose outputs are verifications of an argument's correctness.

In what sense are these applications interactive? Theorem provers can be *automated* in that these applications prove theorems automatically, that is, without user input or suggestions regarding the proof itself, when a theorem is input, automated theorem provers produce either some proof of it or some indication of a failure to produce one; in contrast, interactive theorem provers involve user interaction to construct the proof, so that the user and application collaborate on creating a proof for a given theorem.[7] In fact, interactive theorem provers can (and quite a few do) include automated theorem proving components and tactics for proving theorems; the motivation for this is partly to spare a human reasoner the tedium of checking small or obvious steps that might involve, say, term rewriting or checking that something is a tautology.[8]

Looking underneath the hood, how does an interactive theorem prover verify an argument? Regardless of whether a step is suggested by a user or the application, how does it verify the correctness of the steps? These applications check arguments against their *kernel*, or their core system.[9] Once an argument is formalized within an interactive theorem prover, the application takes that input and creates a corresponding proof object, which is then passed along to the kernel. An interactive theorem prover then verifies whether this proof object can be generated using the core system's rewriting, reduction, and inference rules. To put it in terms more familiar to philosophers, the kernel is analogous to the meta-logic for an interactive theorem prover, and questions of verification are always ultimately passed back to that meta-logic. Of course, this would be no advance at all—it would not boost our confidence in the correctness of the argument with which we started—if the interactive theorem prover had a large or complicated kernel. Hence, best practices require a kernel to be very small so that it can be checked by humans (Geuvers 2009,

[6] Here I am leaving room for axiomatic arguments that do not quite fill in every step. Although it is often controversial whether an argument in a text really does contain gaps—see the criticisms of Reed (2005) towards Hilbert (1899) for instance—examples of such arguments arguably are found in Euclid's *Elements*, Spinoza's *Ethics*, Newton's *Principia Mathematica*, or Whitehead and Russell's *Principia Mathematica*. If the varying sorts of argument found in these texts are all to fall under the phrase 'axiomatic arguments', then it would be overly restrictive to confine the phrase to derivations or proofs in the logical sense.

[7] See (Geuvers 2009, 3) and (Harrison et al. 2014, 1).

[8] The interactive theorem prover *Coq* for example has the CoqHammer automated reasoning tool that searches for a proof of some goal using previously proven (user-provided) lemmas and rewriting available from databases. See https://coqhammer.github.io/ for further details.

[9] The *kernel* is the trusted core of an interactive theorem prover; it passes user commands inputted on the application program interface through the type checker. The kernel typechecks user declarations and rejects inappropriate ones. Much like the kernel of an operating system, the interactive theorem prover's kernel controls the entire program application.

6).[10] 'Small' in this context means a few thousand lines of code; for example, *HOL Light*'s kernel is about 700 lines of code; *Coq*'s kernel is about 14,000 lines.[11] Some interactive theorem provers also have active communities, so that some dozens of people, if not hundreds, have reviewed an interactive theorem prover's small kernel and found it manifestly correct.[12]

Further, interactive theorem provers are *modular* in the sense that users can build theories that are checked by the interactive theorem prover and then can be used to further develop mathematics. For example, one could build a classical propositional logic, check its correctness using an interactive theorem prover, and save this formalization as a standalone module. In building a modal logic, one can then import that classical propositional logic module and extend it with modal operators, new axioms, and a different semantics instead of having to redevelop propositional logic. And users can share their modules with others. The results are that, firstly, any user can develop results and contribute their module—both its code and a summary of what is proven—and, secondly, most interactive theorem provers now have large standard libraries of modules that have been checked by many of the same developers who coded and checked the interactive theorem prover's kernel.[13] The modularity of interactive theorem provers combined with the free availability of many standard libraries make it easy to skip formalizations of base theories by importing standard libraries (though one always can develop base theories from scratch).

These two traits of interactive theorem provers—a small, trustworthy kernel and the advanced libraries that can be imported as standalone modules—are a potent combination. The small kernel through which all advanced development in libraries is automatically passed enables users to be as confident in complex developments as they are in the much smaller kernel. Ill-formed strings, inconsistencies, typing errors, and other mistakes in the development of some more advanced theory return an error message when passed back to the kernel. This allows that mistakes can be corrected as one proceeds. If an inconsistency is derived without an error message, this can be traced back to

[10] Some interactive theorem provers take the further step of creating a proof object that can be verified by type-checking application files apart from the interactive theorem prover's logical system. Those that have this feature satisfy what has become known as the *de Bruijn criterion* following (Barendregt and Barendsen 2002, 323). Although de Bruijn did not coin that term for this property, de Bruijn initiated the *Automath* project, and one design principle for this project was that the software satisfy the de Bruijn criterion.

[11] For comparison, a typical iPhone application has 50,000 lines of code; a Boeing 787 has 6.5 million; a Chevy Volt has 10 million; and an Android OS has 12-15 million (McCandless 2015).

[12] As of April 2021, 3,253 users have starred *Coq*'s Github repository and 194 people have contributed to it. And 376 authors have contributed to *Isabelle*'s Archive of Formal Proofs. The examples throughout this paper usually involve *Coq* or *HOL* descendants like *HOL Light* and *Isabelle/HOL*. These are some of the most popular interactive theorem provers and have been used in the significant applications to higher mathematics discussed below. It should be noted, however, that there are many different interactive theorem provers available and they are widely used in a variety of mathematical applications. A recent and comprehensive critical survey of 41 such applications is given by Saqib Nawaz et al. (2019).

[13] *Coq*'s index of libraries has 579 entries. *Isabelle/HOL*'s Supplemental Library has 147 sections.

some flaw in the kernel. Any bugs can then be removed from that small bit of code in the kernel, and whatever the needed modifications can then be made to save the substantial edifice built upon the kernel.[14]

These combinations have led to recent successes in higher mathematics that leveraged interactive theorem provers.[15] Some much-discussed examples include the proof of the Four-Color Theorem formalized in Coq by Gonthier (2008, 1382), the proof of Kepler's Conjecture formalized in *HOL Light* and *Isabelle* by Hales et al. (2015, 1), and the proof of the Odd Order Theorem formalized in *Coq* by Gonthier et al. (2013, 1).[16] To focus on the last example, the formalized proof of the Odd Order Theorem takes "150,000 lines of proof scripts, including roughly 4,000 definitions and 13,000 theorems," plus "40,000 lines of formal proof." (Gonthier et al. 2013, §6) This formal computer-checked proof is known to be correct on the assumption that *Coq*'s kernel of 14,000 lines does not allow the derivation of an inconsistency.

Naturally, this sort of boost to our confidence in the correctness of a proof is a significant epistemic benefit of working with interactive theorem provers. Using such applications practically brings about a greater degree of exactness and rigor in argumentation, given the error-checking procedures and the smallness of the kernel.[17] Indeed, some involved in successful applications of interactive theorem provers have claimed that such formalized proofs are much more reliable than those checked in the traditional way by humans in the peer-review process. Hales et al. (2015) wrote, "A formal proof in the HOL Light system is more reliable by orders of magnitude than proofs published through the traditional process of peer review."[18]

Despite the boost to one's epistemic confidence that a computer-checked proof can bring, interactive theorem provers are not used by the majority of mathematicians, at least not yet. The reason for this, as Geuvers (2009, 3-4) notes, is that the formalization process involved in applying interactive theorem provers is laborious; if one already believes a pen-and-paper proof—or

[14] The 49 *Coq* modules distributed with the usual `opam` package manager have 383,500 declarations (Müller et al. 2019, 172).

[15] This is not to imply that interactive theorem provers are new. They have been around since the 1960s. A history of such applications is given by Harrison et al. (2014). See also (Geuvers 2009, §2) and (Maric 2015, §2).

[16] See (Maric 2015, §5) for other notable examples.

[17] Indeed, the late Vladimir Voevodsky's work on Univalent Foundations brought fresh focus, energy, and publicity to the case for use interactive theorem provers (Voevodsky 2015, 1278). A concise overview is given in (Awodey et al. 2013, 1165-1166).

[18] "...we have written a *formal proof script* that covers both the mathematical and computational parts of the proof. We have run this script through the Coq proof checking system, which mechanically verified its correctness in all respects. Hence, even though the correctness of our proof still depends on the correct operation of several computer hardware and software components (the processor, its operating system, the Coq proof checker, and the Ocaml compiler that compiled it)...All of them...can be (and are) tested extensively on other jobs, probably much more than the mind of an individual mathematician reviewing a proof manuscript could ever be." (Gonthier 2005, 2)

LaTeX-and-screen proof—is correct, then formalization may seem like a waste of time and energy.[19]

Further, while interactive theorem provers are superb tools for recording proofs and checking them, they are, at least given present technology, less useful for explaining why a theorem holds and for disseminating that result to the broader scientific and public community (Geuvers 2009, 7). This is because mathematicians often draft proofs at a high level, and in a natural language, so that formalizing all of one's scratch work would be cumbrous. And since a majority of mathematicians do not at present use interactive theorem provers, most mathematicians are not prepared to read the application's code.[20] As a result, dissemination of new proofs in interactive theorem provers often proceeds by informal summaries in various publications.

So interactive theorem provers have their limits. Most scientific and public readers do not have facility reading files written within interactive theorem provers. This raises the question, 'Why should the effort be made to formalize an argument in an interactive proof assistant if few colleagues will understand the formal parts and will only read an informal summary?' Furthermore, the work of formalizing arguments raises methodological concerns for historians of philosophy. One could hold, as I do, that interactive proof assistants are not a neutral tool for understanding an argument; rather, formalization using these computational tools presupposes a specific interpretation of the argument, and possibly limits the range of permissible ones to those that can be parsed in the interactive proof assistant's language.

Both the general concern about interactive proof assistants in any field, that the effort may not be worth the gain, and the methodological concerns specific to history of philosophy, that interactive proof assistants are at best not neutral and at worst problematic, may raise for the reader some reasonable doubts against my claim that interactive proof assistants can be useful tools in doing history of philosophy. In §3 I address the methodological concerns that historians of philosophy may have about utilizing interactive proof assistants. In §4 I show that utilizing interactive proof assistants has already been useful to historians of philosophy by discussing examples. Further, in the course of considering these applications, I also argue that formalizing arguments using interactive proof assistants is worthwhile for historians of philosophy for at

[19] However, there are efforts to make using interactive theorem provers less laborious. For an example of some current work in this direction, see the Matryoshka Project website: https://matryoshka-project.github.io/.

[20] As Harrison (1996, §7) notes, there are two styles of input language for interactive theorem provers. The *procedural* style involves giving the application fully explicit instructions regarding what to do next. The *declarative* style involves giving the application a more general direction for what result to establish and perhaps an indication of a strategy for establishing it, but lets the application make explicit the needed steps. One might think of interactive theorem provers in the procedural style as sitting on an opposite pole from the automated theorem provers, whereas those in the declarative style are somewhere in between. Note that one can produce a proof in either the procedural or declarative style in some interactive theorem provers (Harrison 1996, §4). (Harrison et al. 2014, 39) rightly note that most users are better prepared to read the declarative style, particularly those unfamiliar with using interactive theorem provers.

least two reasons. Firstly, if the code utilized in formalizing arguments using interactive proof assistants is *open source*—freely available and editable—then one's work can be further manipulated to create new readings of arguments. Secondly, that concerns about inexplicit assumptions are removed once the argument is computer verified in an interactive theorem prover. One may of course dispute the assumptions as false or distorting to the original argument or text; still, they will be made explicit, and that is a definite advance.

3 Formalization as done by historians of philosophy

In the previous section I explained at some length what an interactive theorem prover is and how they operate. This explanation makes clear some of the following features of formalization using interactive theorem provers:

- There is a learning curve with interactive theorem provers; one must learn how to translate natural language arguments into the interactive theorem prover's system.
- There is a kernel in the interactive theorem prover through which arguments are verified; this kernel is small so as to boost our epistemic confidence in the argument's validity, potentially to a great degree, particularly if the argument is quite complex or, as when one checks a whole theory, many arguments are involved.

However, philosophical historians could fairly raise deep methodological concerns here about the practice of using interactive theorem provers. Indeed, in the title of this article I suggest that computer verification is 'for' historians of philosophy, but one might worry that historians of philosophy do not want it or have good reason to reject it: using interactive theorem provers, it might be thought, are inapplicable to most work done by historians of philosophy, could substantially distort our understanding of an argument, and would be more work than is necessary or helpful, especially as compared with formalization without interactive theorem provers.

To address such concerns, I split them into three grades of involvement: (1) worries about the varieties of work done in history of philosophy that computer verification leaves out, (2) worries about formalization generally and its capacity to distort the arguments of past philosophers, and (3) worries about formalization using interactive theorem provers as compared with formalization without them. These three sorts of concern might be put as follows:

1. There is a great deal of the philosophical historian's work besides formalizing arguments; this work cannot practicably be done with interactive theorem provers.
2. Formalization is not neutral, but partly interpretative; so, formalization in an interactive theorem prover is a potentially distorting influence on philosophical historians seeking to understand the arguments of past figures.
3. Formalization in interactive theorem provers, especially as compared with the work already involved in formalization without using theorem provers, demands much more work than its philosophical payoffs warrant.

Below I address these concerns in numerical order. Beginning with concern (1), there is a rich collection of activities that fall under the umbrella phrase 'history of philosophy.' In a rich essay, Lapointe and Pincock (2017, 13-15) list no less than six "central" activities undertaken by historians of philosophy, and caution that this list is "presumably not exhaustive":

- Rational reconstruction
- Contextualization
- Doctrinal history
- Disciplinary history
- Thematic investigation
- Genealogical narrative

As an example, consider this question: to what extent did Bertrand Russell, as a result of his experiences during World War I, think that his logical orientation in philosophy had ethical implications or think that it was important to develop such ethical upshots of his logical point of view?[21] This question definitely falls within the scope of contextualization, though it could fall within multiple categories depending on the context.[22] How would the use of interactive theorem provers help with contextualization and other activities on the above list? If interactive theorem provers are impractical for these activities, then how can I credibly speak of computer verifications for historians of philosophy—taking historians of philosophy as a group to be engaged in at least all of the activities on the above list?

The answer to concern (1) is of course that interactive theorem provers are not designed to help with understanding the context surrounding the production of texts. Interactive theorem provers are designed to help prove theorems. Accordingly, they can help directly with activities falling under rational reconstruction—specifically, with the interpretation and formalization of arguments—but at most indirectly with other tasks. To fully contextualize Russell's ethical views and writings against the background of his logical philosophy and the events of his lifetime, a historian of philosophy may need to consider Russell's letters, pocket diaries, corpus, personal life, and social connections over the relevant period. Formalizing a specific argument helps with these tasks at best indirectly, through formalizing an argument for or against a particular answer to the contextualization question in the preceding paragraph, although cases where interactive theorem provers actually help in this way are likely to be rare: so far as I know, there is not one such case. And even so-using interactive theorem provers does not show that the premises in one's argument for an answer to this question are supported by the evidence.

[21] For discussion, see (Russell 1914/1986, 55-56), (Klein 2020), and (Elkind 2021).

[22] "The aim of historical contextualization consists in providing an interpretation of philosophical theories and the questions that they are supposed to answer that allows the reader to track the author's philosophical intentions, taking into consideration the relevant aspects of their social, cultural and intellectual environment. In particular, it seeks to determine the role played by previous writings, events or situations in the production of the texts under consideration." (Lapointe and Pincock 2017, 14)

However, none of this weighs against the potential and usefulness of deploying interactive theorem provers in other activities undertaken by historians of philosophy. And none of these activities has to be undertaken in isolation from the rest. Indeed, one's contextualization of a philosopher's views—say, of Russell's views of belief as reacting to William James' view—could lead one to a particular rational reconstruction of Russell's argument against James' position in the 1918 logical atomism lectures. Upon formalizing this argument in an interactive theorem prover, one might discover that it leads to an inconsistency, perhaps with other claims that Russell makes elsewhere. This would suggest indirectly that the contextualization had perhaps missed something relevant to the production of the logical atomism lectures. In this sort of way and in many others, the six activities undertaken by historians of philosophy can and probably should be mutually supporting.[23] So the fact that interactive theorem provers are probably not directly useful in most activities undertaken by historians of philosophy does not lead to the consequence outlined in concern (1) that the usefulness of interactive theorem provers is thereby undermined; rather, because interactive theorem provers are directly useful in rational reconstruction, they can be indirectly helpful in the other areas of concern to historians of philosophy.

Concern (2) amounts to the worry that interactive theorem provers, and even formalization of arguments more generally, might lead to distortions of what past philosophers thought. It is easy to see how this might actually transpire. For example, consider a view of those called "Late-Learners" in Plato's *Sophist*.[24] In that dialogue the Visitor from Elea describes them as holding that no two beings with properties, can ever be mixed together, so that uttering statements which apparently to mix two beings, like 'Socrates is snub-nosed,' is forbidden; one may only say of a good thing g that g is good, or of a human h that h is human.[25] The argument for this prohibition on mixing beings in predication seems to be that "it's impossible for that which is many to be one and for that which is one to be many." Suppose now that we attempted to formalize an argument for their view involving quantifiers over predicates (indicated by capitalized letters) and individuals (indicated by lowercase letters), Quine's device of corner brackets '$\ulcorner$' and '$\urcorner$' to indicate quasi-quotation, 'ϕ' for an arbitrary well-formed formula, and deontic logic's

[23] As Michael Beaney (2013, 253) puts a similar point in terms of his notion of *dialectical reconstruction*, which phrase helpfully "suggests the interplay between rational and historical reconstruction that must continually go on in doing good history of philosophy." Michael Kremer (2013, 311) also puts a similar point in terms of his notion of *philosophical history*, whose goal of "the present *philosophical* understanding of its practitioners" is achieved "*through* understanding the philosophical past..."

[24] Since this example is for purposes of illustration, I will not venture into scholarly controversies over what the view actually is or who, if anyone, may have held it.

[25] "They evidently enjoy forbidding us to say that a man is good, and only letting us say that that which is good is good, or that the man is a man." (Plato 1997, 251b; see also the following passages)

impermissibility operator 'IM' (without taking a firm stance on what kind of impermissibility the Late-Learners meant):[26]

1. $\neg\exists F\exists G\exists x[(Fx \wedge Gx) \wedge F \neq G]$.
   It is impossible for that which is one thing to be many in its characteristics.
2. $\neg\phi \implies \mathrm{IM}[\mathrm{speak}(s, \ulcorner\phi\urcorner)]$
   It is impermissible to speak falsely.
3. $\mathrm{IM}[\mathrm{speak}(s, \ulcorner\exists F\exists G\exists x[(Fx \wedge Gx) \wedge F \neq G]\urcorner)]$
   So, it is impermissible to speak so as to attribute distinct properties to a thing. (By (1) and (2), substituting (1) for ϕ.)

The Late-Learners would object to this way of formalizing their argument because it already mixes different beings together in premise (1).[27] This example illustrates the general point in concern (2) that formalization, whether in interactive proof assistants or not, could distort our understanding of a past philosopher's argument or view.

Concern (2) can be answered by pointing out that formalization in general is not better or worse off than other modes of interpretation. The example above gives an argument for the Late-Leaner's view formally and informally. Both are interpretations of their position in the sense that each provides reasoning for their conclusions. Both can be criticized as being dissonant with the Late-Learner's other views or methods, being historically inaccurate or ill-fitting the dialectical or historical contexts, or as being directly inconsistent with their thesis that beings with different characteristics cannot be mixed. Indeed, as Castagnoli (2010, Chapter 13) reads Plato, this is just the criticism of the Late-Learners that Plato develops: no argument is consistent with their thesis because, if true, their view forbids any making predications at all. If that is right, then any rational reconstruction or interpretation of their view that provides an argument for it violates the constraints that accepting their view imposes.

So concern (2) only arises through a misunderstanding: nobody should maintain that formalization is not interpretative. But like any (even non-formalized) interpretation, formalized arguments found in past philosophers' works will need correction and refinement. This can come from alternative formalizations, informal presentations, providing more context, and all the other sources upon which historians of philosophers draw to correct and refine informal rational reconstructions of arguments.

Indeed, Fuenmayor and Benzmüller (2018a, 2019a, 2020) have already developed a pioneering framework for conducting interpretations using interac-

[26] The logic of development here is assumed to be a classical logic extended by the modal operator 'impermissible' and with the quantifiers ranging over (at least) everything in the spatial-temporal universe (and their properties). It is worth bearing in mind that formalization is always formalization in some logical system, but as we will see, the Late-Learners would seemingly be opposed to formalization within any system (unless one could be devised that is free of predication period).

[27] Indeed, the Late-Learners arguably would reject any formalization, or even natural language formulation, of an argument for their view. The Visitor argues in fact that the Late-Learner's view cannot be stated because any predication is impossible on their view, at least according to Castagnoli (2010, Chapter 13).

tive theorem provers to formalize natural language arguments generally. Fuenmayor and Benzmüller (2018a, 2) call their approach *computational hermeneutics* and describe their framework as "a novel approach to the logical analysis (aka. formalization) of arguments" while criticizing the perhaps common view that formalizing natural language arguments is "a kind of artistic skill that cannot be standardized or taught methodically".[28] Part of the reason they use the word 'hermeneutics' is that they adopt, for methodological reasons, the notion of a *radical interpretation* from Donald Davidson (1973, 1994).[29] Setting aside issues of interpreting Davidson's views, Fuenmayor and Benzmüller (2018a, 3) read Davidson as holding that any interpretation another person's argument must indicate the "compositional structure" of their language and be capable of being "assessed using the evidence available to the interpreter."[30] Fuenmayor and Benzmüller (2018a, 3) meet these requirements by indicating that a Tarski-style characterization of truth in a language would indicate a language's compositional structure and by noting that intersubjectively observable events in the world, including further speech acts, would allow for assessing interpretations. Then (Fuenmayor and Benzmüller 2018a, 4) say further that an interpretation of a sentence or argument should also be assessed within the whole network of the argument or arguments studied. As Fuenmayor and Benzmüller (2020, 196) hold that such assessment and reinterpretation proceeds through "an iterative process of 'trial-and-error'" wherein interpretive formalization and checking is done and redone "at least several hundreds of times" to reach a satisfactory interpretation of given argument. Hence the appropriateness of their calling it 'hermeneutics.' The 'computational' part of course comes from their use of interactive theorem provers in this process, and they describe in helpful detail how the interactive theorem provers assist in doing interesting interpretative work; see especially (Fuenmayor and Benzmüller 2019a, Figures 4 and 9).

We need not adopt this computational hermeneutics framework, or, say, Davidson-inspired views on radical interpretation and meaning for example, to appreciate the following upshot of Fuenmayor and Benzmüller's pioneering work: formalization with or without interactive theorem provers can be a useful mode of interpreting a philosophical text, just as informal argument

[28] This name is slightly unfortunate because 'computational hermeneutics' had some years earlier been used to describe hermeneutic practices involving very different computational tools (Harnad 1990). Since this alternative usage seems to be well-established among digital humanists (Mohr et al. 2015; Rockwell and Sinclair 2016; Piotrowski and Neuwirth 2020), and it predates the newer usage by over twenty years, I will only use 'computational hermeneutics' in describing Fuenmayor and Benzmüller's methodology.

[29] As Michael Beaney (2013, 247) has noted, Richard Rorty (1984, 52-53, footnote 1) talks of a hermeneutic circle with respect to the relationship between rational and historical reconstruction. This additional layer of nuance seems consistent with the computational hermeneutics framework.

[30] Note that by "compositional structure," they mean this primarily in the Tarskian sense that a recursive specification of the truth evaluation of formulas should be available. They do not mean "compositional structure" in the Fregean sense that the meaning of a formula is built up from the meanings of its parts, that is, they do not endorse what Szabó (2020, §1.6.5) calls "the building principle."

reconstructions can be. Computational hermeneutics is simply a thoroughly worked out model of how interactive theorem provers can be used in interpreting texts, and one which shows an impressive amount of sensitivity to issues of methodology. It is in short a framework that addresses concern (2) in showing by example that formalization is a rich, not limiting, mode of interpretation.

As an aside, it should be further noted that formalization in a specific theorem prover, using its particular formal system, does not constrain the field of possible interpretations to only its specific system. This is because some theorem provers allow for what is called an *embedding* of various logical theories, including their axioms and semantics, using the theorem-prover's language and logic: just as interactive theorem provers can prove theorems about geometric objects like triangles and squares, it can prove theorems about logics like a free logic, a modal one, or a set theory like Zermelo-Fraenkel (Harrison et al. 2014, 20-21). Logics themselves can be taken as objects of study and facts can be proven about or within them. All of this still would be passed back through the kernel, so that the interactive theorem prover still verifies the argument, but the effect of passing the embedded logic and proof within it back through the kernel would be to verify the argument from within the object logic. The situation here is rather like ours when we study Mars from the Earth: we do not need to abandon our planet in studying the Red Planet from afar, and unless something ruined the security of Earth, there is no reason why we cannot verify features of Mars, or establish what living there would be like, from our Blue Planet. The interactive theorem prover is like the Earth in this analogy: from its secure place, we can study or model the internal or external features of practically any object logic as we please.

Finally, we reach concern (3). Why should we use interactive theorem provers when formalizations in other media—pen and paper, marker and whiteboard, or software applications like Word or TEX—would serve many of these same purposes? Is it not a great deal of extra work to learn how interactive theorem provers work, how to use them, and how to share results in these applications, without much added payoff?

The answer to concern (3) is similar to what I said in response to concern (2). No one must use interactive theorem provers against their wishes. Indeed, no one must use applications like Word or TEX: a rich or privileged person could handwrite or dictate their thoughts and pay someone else to digitally transcribe and format them for journals or publishers. Interactive theorem provers stand to other modes of formalization as applications like Word and TEX stand to pen and paper.[31] In particular, using interactive theorem provers, like using typesetting applications like Word or TEX, is not required to do history of philosophy, but these tools can be very useful and are found to be so by many of those who use them: the examples cited in §1 above and those

[31] This is not to say the programs are exactly analogous; Word and TEXdo not solve logic problems or engage in heuristic search as interactive theorem provers do. But both sorts of software applications are alike in this respect: they are both technologies that greatly assist with specific tasks (though not without certain costs), and history of philosophy was done (and still could be done in principle) without them.

discussed in §4 below show this incontrovertibly. We can and should admit this without, as a discipline, making mastering interactive theorem provers—or typesetting programs—a prerequisite to doing history of philosophy.

Further, using interactive theorem provers in formalizing arguments brings some benefits not brought by formalization in other media. For example:

- Once an argument is formalized in an interactive theorem prover, anyone with the formalization sources can automatically check its validity on their own computer by running its formalization sources through the application (though one cannot automatically check that it is a faithful reconstruction). In contrast, checking an argument's validity in other media requires a human-reader's time and attention.
- Libraries of theories with definitions, theorems, and proofs have been developed by user communities for interactive theorem provers. These libraries can be imported into one's own file with relative ease so that one does not have to redo their work. In contrast, to import another person's proofs, definitions, or theorems in other media, one has to laboriously and tediously recopy their work (assuming one has permission to do so), or be content with just citing their developments.
- The formalization sources of an argument in an interactive theorem prover can be freely and widely shared (indeed, there is a norm of sharing such code among programmers and in the open science movement more broadly). This allows others to freely modify one's formalization to create a new interpretation that builds on what someone else has done before. In contrast, publishers using many other modes of publication and dissemination, for good reason, lock their disseminations of works containing formalizations in non-editable formats (though sharing the Word or TEX source file bears some benefits similar those that come from sharing the formalization sources for interactive theorem provers).

So in response to concern (3), nobody has to use interactive theorem provers against their wishes, but those who have used them report finding them beneficial and useful for some purposes. Further, using them brings some benefits that formalization in other media does not bring, and some other benefits brought by any media such that source files are sharable but not other media. So concerns (1), (2), and (3) do not weigh against deploying interactive theorem provers in doing philosophical history.

On the other hand, the best response to skepticism about the value of something is often an example, or many examples, that show its value. In this case, skepticism about the value of interactive theorem provers in doing philosophical history can be best addressed by giving examples of them being used in interesting philosophical histories: the best answer to the question 'Can computer proof-assistants serve as a tool for philosophizing?' is 'Yes, they already do.' In §4 I describe multiple examples to show that this is so.

4 Examples of computer verification in history of philosophy

In §3 I discussed and addressed the reasonable methodological concerns that historians of philosophy may have about using interactive theorem provers. But the most effective way to settle doubts about the usefulness of new technologies in an old endeavor is to give an example. Incidentally, many examples of philosophical history produced with interactive theorem provers were given in §1: these samples span from Aristotle's syllogistic to Leibniz's account of concepts to Newton's *Principia* to twentieth-century ontological arguments and deontic logics. Though space precludes a comprehensive discussion of all these examples, below I give a representative sample that shows the many ways that interactive theorem provers have shown themselves to be useful in doing history of philosophy.

The first example discussed will be the recent application of *Coq* in reconstructing complete proofs from the demonstrations in Whitehead and Russell's *Principia Mathematica*. This will serve as a helpful illustration of how interactive theorem provers can be (and now are) used to reconstruct arguments from a text. A reader may of course worry that applying interactive theorem provers to formalized (or even quasi-formalized) texts does little to show that such applications can be (and now are) useful in reconstructing arguments made in some informal language. The other two examples discussed below involve reconstructing and formalizing arguments from non-formalized texts that nonetheless show some benefit to applying interactive theorem provers in doing history of philosophy. The second example (in ancient philosophy) is a formalization of Aristotle's syllogistic; the third example (in ethics) is a formalization of Gewirth's argument for the principle of generic consistency. These other two examples thus address the concern that interactive theorem provers would only be useful with formalized or quasi-formalized texts.

4.1 Applying interactive theorem-provers to *Principia Mathematica*

First I will discuss in detail a recent application of interactive theorem provers to the propositional logic of Whitehead and Russell's *Principia*.[32] Discussion of this example will indicate the usefulness, though not indispensability, of interactive theorem provers in history of philosophy. First, I describe what was done in this application. Then I discuss what was learned from it.

In a sentence, the entirety of *Principia*'s propositional logic ($*1 - *5$) was computer verified in the interactive theorem prover *Coq*. *Principia*'s propo-

[32] One might be worried about the paradox reported in (Kirchner et al. 2020, §5). The paradox arose there within the context of a shallow embedding of abstract object theory, a hyperintensional second-order modal logic founded upon the logic of relations, within an extensional higher order logic founded upon the logic of functions. The paradox that arises, however, is due to the formation of complex terms using λ-expressions and definite descriptions; *Principia*'s simply-typed grammar does not permit the comprehension of universal properties, but only allows the formation of terms that are properties of all members of some universal class through its comprehension schema $*12$.

sitional logic has 206 starred numbers all-told: 10 are primitive propositions, 7 are definitions, and 189 are theorems. This takes just 36 pages in its first edition As O'Leary (1988, 108) says, "The work done in the propositional logic section of *Principia Mathematica* is a model of succinct expression." In a model of succinct computational expression, coding all of *Principia*'s propositional logic in *Coq* takes just 4,360 lines of code.

It is important to note that *Principia*'s 'proofs', for the most part, are actually demonstrations, meaning they are incomplete sketches of proofs intended to be sufficient for the reader to reconstruct complete proofs.[33] A comparison to a cookbook is helpful here: *Principia* gives readers demonstrations, which are like recipes for cooking up the complete step-by-step proof, but the cooking is left to the reader. So what it means for a formalization to successfully reconstruct the reasoning in *Principia* is that it reconstructs a proof that follows the demonstration's recipe: one has to produce a complete step-by-step proof that (a) cites every theorem mentioned in the demonstration and (b) only uses axioms, rules of inference, or theorems previously laid down or proved in *Principia*. This is precisely what was achieved: in every case we gave a complete step-by-step proof that cited every theorem mentioned in the proof-sketch, and we never departed from the logic of *Principia*.

In short, the encoding of *Principia*'s propositional logic in *Coq* verified not just that its theses are theorems, but that each one can be proved *according to the proof recipe given in the text*. A result of doing this was a freely available source file that is now posted in a *Github* repository.[34] So anyone who wishes to verify the propositional logic demonstrations of *Principia* needs only to download *Coq* and run the source file in the interactive theorem prover. They can even modify that formalization sources as they like to investigate redundancies, alternative proofs, and so on.

Another result of this encoding was the discovery of some statistically interesting information about the omissions of steps in *Principia*'s proof recipes. Recall that the proof recipes omit steps. But what has never been explicitly shown, or discussed in more than passing mention, is that Whitehead and Russell omitted steps systematically: they omitted 'easy' steps involving applications of definitions, commutations, associations, and double negations. Some summary data is given in Table 1.[35]

As this data shows, most often *Principia*'s demonstrations omit mention of inference rules like ∗1·1·11 and ∗3·03, or applications of definitions like ∗1·01 and ∗4·01, that were omitted from demonstrations. Such substantive omissions were very rare. The general pattern was to omit principles of infer-

[33] One reason given for this in *Principia* is to abbreviate tedium (Whitehead and Russell 1910, 93, 96). Another plausible reason, not mentioned explicitly in the work, is mitigating the cost of printing, which was substantial to the authors—each author paid £50 in 1910, which is about £6,022.92 in 2020 terms (€7090.77, or US$8,086.50, or C$10,506.17), according to the Bank of England's inflation calculator: https://www.bankofengland.co.uk/monetary-policy/inflation/inflation-calculator.

[34] Link omitted for purposes of blind review.

[35] Substitutions were the most common omissions because substitutions into axioms or theorems used as lemmas occur in practically every proof.

Principles	Kind of principle	Omissions
Substitution of formulas into theorems	Specialization	184
∗1·1/∗1·11	*modus ponens*	101
∗1·01/∗·3·01/∗4·01	Definition	78
Substitution of material equivalents	Replacement	59
∗2·05/∗2·06	Syllogism	46
∗3·22/∗4·21/∗4·3/∗4·31	Commutation	28
∗2·31/∗2·32/∗2·33/∗4·32/∗4·33	Association	12
∗2·12/∗2·14/∗4·13	Double negation	11
∗2·02/∗3·26/∗3·27	Simplification	8
∗3·3/∗3·31/∗4·87	Exp/Importation	6
∗2·03/∗2·15/∗2·16/∗2·17/∗3·37/∗4·1/∗4·11	Transposition	4
∗1·2/∗1·3/∗1·4/∗1·5/∗1·6	Axiom	1

Table 1 The (systematic) omissions in *Principia*'s propositional logic proof recipes

ence or leave unspecified the needed substitution into a cited theorem. Most omitted theorems were replacements involving double negation, commutation, or association—the 'easy' steps.

Besides this data, another result was that, for the first time, we have for every theorem a full step-by-step proof that (a) follows *Principia*'s proof recipe and (b) only uses principles previously given as axioms or theorems in *Principia*. This encoding in *Coq*, in contrast with earlier applications of computers to *Principia*'s logic, thus allows us to reconstruct the complete reasoning indicated in the text.

Of course, the first few proofs in *Principia* are short and *Principia* gives them in full detail (Whitehead and Russell 1910, 102). But because *Principia* mostly gives demonstrations, many proofs in the *Coq* encoding do not correspond one-to-one with what *Principia* describes. Take, for example, *Principia*'s brief proof-recipe for ∗2·37:

∗2·37. $\vdash :: q \supset r . \supset : q \vee p . \supset . p \vee r$ [Syll . Perm . Sum]

There is no indication of what uniform substitution is required (if one is needed) in the three starred numbers are cited as lemmas. Further, 'Syll' is polysemous in *Principia*: four different theorems are called 'Syll' and the demonstration does not indicate which of them are used. As we found in reconstructing the full proof in *Coq*, the proof recipe calls for one substitution into Syll(∗2·05) and another into Syll(∗2·06) plus two different substitutions into Perm(∗1·4):

```
Theorem n2_38 : ∀ P Q R : Prop, (Q → R) → ((Q ∨ P) → (R ∨ P))
Proof. intros P Q R.
        specialize Perm1_4 with P R.
        intros Perm1_4a.
        specialize Syll2_05 with (Q∨P) (P∨R) (R∨P).
        intros Syll2_05a.
        MP Syll2_05a Perm1_4a.
        specialize Perm1_4 with Q P.
```

```
        intros Perm1_4b.
        specialize Syll2_06 with (Q∨P) (P∨Q) (P∨R).
        intros Syll2_06a.
        MP Syll2_06a Perm1_4b.
        Specialize Sum1_6 with P Q R.
        intros Sum1_6a.
        Syll Sum1_6a Ha Sb.
        exact Sb.
Qed.
```

This proof is faithful to the demonstration in this precise sense: it cites every theorem mentioned in the recipe and nothing beyond the development given earlier in *Principia*.[36] Why should a historian of philosophy bother reconstructing the full proof? Because doing so allows one to understand the reasoning indicated in the text. A reconstructing of the full proof in *Coq* can be used to reconstruct the full proof in *Principia*'s system. For ∗2·37, the full proof is as follows:

∗2·37. $\vdash :: q \supset r \,.\supset: q \vee p \,.\supset.\, p \vee r$

Dem.

$$\left[\text{Perm}(*1{\cdot}4)\ \frac{q,\ p}{p,\ q}\right] \qquad \vdash : q \vee p \,.\supset.\, p \vee q \tag{1}$$

$$\left[\text{Syll}(*2{\cdot}06)\ \frac{q \vee p,\ p \vee q,\ p \vee r}{p,\quad q,\quad r}\right] \qquad \vdash :: q \vee p \,.\supset.\, p \vee q :.\supset :. \\ p \vee q \,.\supset.\, p \vee r : \supset : q \vee p \,.\supset.\, p \vee r \tag{2}$$

$$[\text{MP}(*1{\cdot}1)\colon (1) \wedge (2)] \qquad \vdash :. p \vee q \,.\supset.\, p \vee r : \supset : q \vee p \,.\supset.\, p \vee r \tag{3}$$

$$[\text{Sum}(*1{\cdot}6)] \qquad \vdash :. q \supset r : \supset : p \vee q \,.\supset.\, p \vee r \tag{4}$$

$$\left[\text{Syll}(*2{\cdot}06)\ \frac{q \supset r,\ p \vee q \,.\supset.\, p \vee r,\ q \vee p \,.\supset.\, p \vee r}{p,\quad q,\quad r}\right] \\ \vdash ::: q \supset r : \supset : p \vee q \,.\supset.\, p \vee r :: \supset :: p \vee q \,.\supset.\, p \vee r : \\ \supset : q \vee p \,.\supset.\, p \vee r :. \supset :. q \supset r : \supset : p \vee q \,.\supset.\, p \vee r \tag{5}$$

$$[\text{MP}(*1{\cdot}1)\colon (4) \wedge (5)] \qquad \vdash :: p \vee q \,.\supset.\, p \vee r : \supset : q \vee p \,.\supset.\, p \vee r :. \supset :. \\ q \supset r : \supset : p \vee q \,.\supset.\, p \vee r \tag{6}$$

$$[\text{MP}(*1{\cdot}1)\colon (3) \wedge (6)] \qquad \vdash :. q \supset r : \supset : p \vee q \,.\supset.\, p \vee r \qquad \text{QED}$$

For readability, I will use '∧' for conjunction in place of *Principia*'s dots for conjunction, adjusting scope dots as needed. *Principia* abbreviates the last three lines of this proof with the derived rule 'Syll' as was done in the *Coq* encoding. Notice how *Principia*'s full proof correlates one-to-one with the step-by-step proof encoded in *Coq*. This marks a faithful reconstruction.

[36] There are many full proofs that are faithful in this sense, and sometimes the shortest proof faithful to the recipe will not be unique. Further, some proof recipes contain steps that are unnecessary.

Another illustrative example is the demonstration of Exportation or 'Exp' (∗3·3). This demonstration cites Syll just once. It turns out that the complete proof involves two theorems named 'Syll'—∗2·05 (`Syll2_05` in *Coq*) and ∗2·06 (`Syll` in *Coq*)—are used:

∗3·3. $\vdash :: p \wedge q \,.\supset.\, r : \supset : p \,.\supset.\, q \supset r$

Dem.

[Id ∧ (∗3·01)]	$\vdash :: p \wedge q \,.\supset.\, r : \supset : \sim(\sim p \vee \sim q) \,.\supset.\, r :$
[Transp]	$\supset : \sim r \,.\supset.\, \sim p \vee \sim q :$
[Id ∧ (∗1·01)]	$\supset : \sim r \,.\supset.\, p \supset \sim q :$
[Comm]	$\supset : p \,.\supset.\, \sim r \supset \sim q :$
[Transp ∧ Syll]	$\supset : p \,.\supset.\, q \supset r :: \supset \vdash .$ Prop

Principia frequently omits one or more theorems needed for a demonstration to omit in the corresponding complete proof, as above. For example, in the demonstration of ∗2·74, three theorems are cited, but the demonstration omits two theorems—∗2·31 and ∗2·32—are needed in the full proof. So some demonstrations omit a theorem found to be necessary in our *Coq* formalization. This is one way that applying interactive theorem provers can facilitate our understanding, as historians of philosophy, of an argument in a text.

Another way that interactive theorem provers can help is by forcing us to follow each step carefully. By computer verifying our inputs for correctness, misprints in demonstrations can be found. Consider the proof of ∗5·23:

∗5·23. $\vdash :: p \equiv q \,.\equiv : p \,.\, q \,.\vee.\, \sim p \,.\, \sim q \qquad \left[*5{\cdot}18 \,.*5{\cdot}22\, \frac{\sim q}{q} \,.*4{\cdot}13{\cdot}36 \right]$

In what appears to be a misprint, *Principia*'s demonstration here cites ∗4·36, which is $P \equiv Q : \supset : P \equiv R \,.\equiv.\, Q \equiv R$. But at this point in the proof, having used all the other cited starred numbers, we already have $P \equiv Q :\equiv: P \wedge Q \,.\, \vee \,.\, \sim Q \wedge \sim P$. The *Coq* proof instead used ∗4·3 to commute $\sim Q \,.\wedge.\, \sim P$ into $\sim P \,.\wedge.\, \sim Q$ as needed.

Notice also that constructing the complete proof of ∗5·23 shows three theorems are required that the demonstration omits—∗1·1 (*modus ponens*), ∗3·03 (*conjunction*), and ∗4·22—and that non-obvious substitutions are required for the four theorems that the proof recipe does mention. We thus understand the text better with the assistance of interactive theorem provers like *Coq*.

Recall the three benefits of utilizing interactive theorem provers in doing philosophical history that were suggested at the end of §3. As mentioned before, the formalization sources are now freely available to anyone who wishes to computer verify the proofs in minutes or to modify the reconstructed proofs according to some preferred interpretation. Recall that the second benefit of utilizing interactive theorem provers was that there are advantages to importing logical developments made in standard libraries rather than needing to copy such work manually or to review it oneself: the computer verification of

Coq can be trusted about as much, if not more so, than any human-checked base logic, and computer checking the library of developments building on that base spares the human reader much time reviewing standard developments.

This application of the interactive theorem prover *Coq* to *Principia*'s proof recipes also shows the advantages of interactive theorem provers' standard libraries. Classical logic libraries were imported so that *Coq* could validate *Principia*'s inference rules, axiom schemata, and definitions (well-formedness rules were not checked because ill-formed strings simply are not processed in *Coq*). All the primitives of *Principia* were thus validated by *Coq*'s (non-classical) logic and its standard libraries for classical logic. Checking these primitives in *Coq* provided extra epistemic support that the system of *Principia* is consistent provided *Coq*'s kernel and standard libraries are so. Also, *Coq* checked that each step and substitution in the full proofs based on *Principia*'s demonstrations are valid steps within *Principia*'s own system.

The upshot of all this is that the axioms of *Principia*, having been proved in *Coq*, do not introduce any inconsistencies unless *Coq*'s kernel and standard libraries allow the derivation of one. The result of this combined with producing complete step-by-step reconstructions of *Principia*'s proofs, which steps were also computer checked, is to boost our epistemic confidence in the full proofs and theorems in *Principia* as reconstructed in *Coq*: their correctness depends only upon our epistemic confidence in *Coq*'s kernel and standard libraries, which is certainly less likely to contain an error than a lengthy and difficult development given in a work like *Principia*. It takes little imagination to consider the extent of the advance that would be achieved by undertaking a similarly complete and computer verified reconstruction of Volumes I, II, and III of *Principia* in their entirety. Likewise for other works: Frege could perhaps have been spared some post-publication logical heartbreak.[37]

This encoding in *Coq* also provides a sharable, editable, and computer verified reconstruction of the reasoning in the text, with the additional upshot of boosting to our epistemic confidence in *Principia*'s demonstrations. This definite advance is made using interactive theorem provers to do interesting work in philosophical history.

4.2 Applying interactive theorem provers to Aristotle's syllogistic

Next I will discuss in detail an application of the interactive theorem prover *Isabelle* in checking a reconstruction of Aristotle's syllogistic.[38] In formaliz-

[37] Curiously, this seems to be an empirical claim. Is there a general-use interactive theorem prover that can detect the inconsistency in Frege's *Grundgesetze* without direct instructions from a human user? Not many attempts to formalize Frege's systems in interactive theorem provers have been made, but this would be a worthwhile test case for an interactive theorem prover's inferential engine.

[38] *Isabelle* is a generic proof assistant. *Isabelle* comes in different instances, that is, the base logic for the theorem-proving environment can differ; the usual instance of *Isabelle* used is *Isabelle/HOL*, which is a higher-order logic theorem-proving environment. Here I will use

ing Aristotle's syllogistic in *Isabelle*, Angeliki Koutsoukou-Argyraki (2019) is verifying the account of Aristotle's syllogistic given by Robin Smith (2020).

Koutsoukou-Argyraki (2019, §1.1) first expresses Aristotle's syllogistic using set-theoretic machinery; Koutsoukou-Argyraki notes that 'Socrates is an animal' and 'humans are animals' both get "the same logical analysis"—'all things identical with Socrates are animals' and 'all humans are animals'—so that each one can be couched in set-theoretic relations. Thus A, E, I, and O propositions are analyzed as follows:

```
definition universal_affirmation :: "'a set  → 'a set  → bool"
        where "A Q B ≡ ∀ b ∈ B . b ∈ A "
```

Any element of B is in A.

```
definition universal_denial ::  "'a set  →'a set   → bool"
        where "A E B ≡ ∀ b ∈ B. (b ∉ A)  "
```

Any element of B is not in A.

```
definition particular_affirmation ::  " 'a set  →'a set  → bool"
        where "A I B ≡ ∃ b ∈ B. (b ∈ A) "
```

Some element of B is in A.

```
definition particular_denial ::  "'a set  →'a set → bool"
        where "A Z B ≡ ∃ b ∈ B. (b ∉ A) "
```

Some element of B is not in A.

Koutsoukou-Argyraki then gives and computer-checks the conversion rules:

```
lemma aristo_conversion1 :
        assumes "A E B" shows "B E A"
        using assms universal_denial_def by blast
```

An E proposition 'No A is B' can be validly converted into 'No B is A.'

```
lemma aristo_conversion2 :
        assumes "A I B" shows "B I A"
        using assms unfolding  particular_affirmation_def
        by blast
```

An I proposition 'Some A is B' can be validly converted into 'Some B is A.'

```
lemma aristo_conversion3 :
        assumes "A Q B" and "B ≠ {}"  shows "B I A"
        using assms
        unfolding universal_affirmation_def
                particular_affirmation_def by blast
```

'*Isabelle*' generically without specifying the instance. More information about *Isabelle* is available at https://isabelle.in.tum.de/overview.html.

An A proposition 'All *A*s are *B*s,' where *B* is non-empty, can be validly converted into 'Some *B* is *A*.' These proofs are of interest partly because they computationally verify the Aristotle's natural language deductions in the text.

The last conversion rule is particularly interesting because *Isabelle*'s automated proof checking detects counterexamples *without* the assumption that *B* is non-empty (Koutsoukou-Argyraki 2019, 2). There has been a long controversy over whether this was an appropriate assumption in logic and how to treat converting A propositions into I propositions along Aristotelian lines; see discussion and references in (Parsons 2021, §1.2). Formalization in *Isabelle* brings out explicitly the existence assumption implicit in arguing, as Aristotle does, for the soundness of this conversion (Smith 2020, §5.2). Indeed, *Isabelle* would flag the fact that this conversion fails if *B* is empty even if a human had missed it. Here is a case where formalization of natural language arguments using interactive proof assistants serves to make explicit tacit assumptions that were the subject of centuries-long controversy.

Koutsoukou-Argyraki (2019, §1.2) then computer-checks Aristotle's 14 arguments in *Prior Analytics* I.4-6 for (as Smith (2020, §5.4) puts it) the premise combinations yielding deductions. These proofs largely follow from the definitions above and applications of *Isabelle*'s theorem-proving tactics *auto* and *blast*. The *blast* tactic applies classical reasoning (much as one might apply natural deduction techniques) to secure the desired goal; the *auto* tactic applies classical reasoning and also attempts simplifies the formula by rewriting.[39] To get a general sense of how these proofs go, here is the AAA syllogism 'All Greeks are humans; all humans are Greeks; so, all Greeks are humans' verified in *Isabelle* Koutsoukou-Argyraki (2019, §1.2.4):

```
lemma GreekMortal :
        assumes  "Mortal Q Human" and "Human Q Greek "
        shows " Mortal Q Greek "
        using assms Barbara by auto
```

Finally, there are meta-theoretic results about Aristotle's syllogistic. As Smith (2020, §5.5) has it, Aristotle shows certain meta-theoretical results in *Prior Analytics* I.4-6, including this:

– All syllogisms reduce to the two universal syllogisms in the first figure.

The two universal deductions of the first figure are AAA-1 (a.k.a. *Barbara*) and EAE-1 (a.k.a. *Celarent*). Following Smith's reading, Aristotle argues in two steps: (1) it is shown by *reductio* that the particular deductions of the first figure (AII-1 and EIO-1, a.k.a. *Darii* and *Ferio*) reduce to universal deductions of the second figure (EAE-2 and AEE-2, a.k.a. *Cesare* and *Camestres*); (2) all particular deductions of whatever figure reduce to particular deductions of the first figure. And Aristotle has previously argued for the soundness of the universal deductions using *Barbara* and *Celarent*. Consequently, all deductions reduce to *Barbara* and *Celarent*.

[39] For more information, see the *Isabelle* reference manual (Wenzel 2021, §9.3-§9.4).

These steps of the argument, and each needed reduction, are computer-checked in *Isabelle* by Koutsoukou-Argyraki (2019, §1.3). This example shows that interactive theorem-provers can be leveraged to (1) verify meta-theoretical results, (2) reconstruct natural language arguments for them, and (3) study logical systems, once suitably embedded, other than the base logic that provides the interactive theorem-prover's theorem-proving environment.

4.3 Meta-ethics in Interactive Theorem Provers

You might worry that the above examples are formalized or quasi-formalized already: *Principia* and *Prior Analytics* are both works in logic, which might lead you to wonder if I have cherry-picked examples that help my case but are not representative.

To answer this objection, and to further address skepticism about the value of applying interactive theorem provers in doing history of philosophy, I next discuss a non-logic example of applying interactive theorem provers to areas outside logic in doing history of philosophy. In this sub-section I discuss an example in ethics due to David Fuenmayor and Christoph Benzmüller. (There are also examples in metaphysics, including formalizing axioms for mereology, ontological arguments, and abstract object theory; see §1 for references.)

The ethical example is Fuenmayor and Benzmüller's formalizing an argument of Alan Gewirth (1981).[40] Fuenmayor and Benzmüller (2018b, 2019b), following the interpretation of Beyleveld (1991), reconstruct Gewirth's argument for *the principle of generic consistency* according to which "agents categorically ought to act out of respect for the rights of all agents." (Beyleveld 2012, 2, note 1) More precisely:

> In a nutshell, according to this principle [the principle of generic consistency], any intelligent agent (by virtue of its self-understanding as an agent) is rationally committed to asserting that (i) it has rights to freedom and well-being, and (ii) all other agents have those same rights. (Fuenmayor and Benzmüller 2019b, 423-424)[41]

To formalize Gewirth's (lengthy) argument for this view, Fuenmayor and Benzmüller (2019b, §3) first encode the broader ethical framework, formalizing the notions of *purpose*, *acting on a purpose*, (subjective) *goodness*, *freedom*, *well-being*, *obligation*, *interference*, and *rights*. For example, a prospective purposive agent is an individual who can act on some purpose. Fuenmayor and Benzmüller formalize this as

[40] Although this might not be an example of doing history of philosophy, Gert-Jan C. Lokhorst (2010, 2011) has leveraged interactive theorem provers (specifically, *Prover9*) in doing meta-ethics; Lokhorst has applied *Prover9* to repairing apparent defects in Mally's formulation of deontic logic and to modeling a robot capable of ethical reasoning. Additional examples are cited in (Fuenmayor and Benzmüller 2019b).

[41] As glossed by Beyleveld (1991, 1), "purposive agents and prospective purposive agents (PPAs) are rationally committed, by virtue of conceiving of themselves as PPAs, to assenting to a moral principle, "the principle of generic consistency" (PGC), which states that all PPAs have claim (or "strong") rights to their freedom and well-being."

```
definition PPA:: "p" where "PPA a ≡ ∃E. ActsOnPurpose a E"
```

where *purpose* has an antecedent definition $e \Longrightarrow m \Longrightarrow m$, which indicates something like the desire of e that m obtain. *Goodness* in Gewirth's view is tied to subjective value and an agent's purpose. So Fuenmayor and Benzmüller (2018b, §3.3) formalize this as

```
axiomatization where explicationGoodness1:
        '' ⌊∀ P. ActsOnPurpose a P → Good a P⌋^D ''
```

which is to say that a thing is (subjectively) good (for an agent) means (by these postulates) that (1) any prospectively purposive agent a acting on a purpose P is such that P is good for a; (2) for any prospective purposive agent a acting on a purpose P and for any M, if M is needed for a to achieve purpose P and P is good for a, then M is good for a; (3) for any state of affairs ϕ and any prospective purposive agent a, if ϕ is possible given purpose P, then any ϕ that is necessarily good for a is also obligatory for a.

Notably, this explanation of *goodness* in Gewirth's argument is that the third condition—that "from an agent's perspective, necessarily good purposes are not only action motivating, but also entail an instrumental obligation to their realization (but only where possible)"—is required to validate the argument (Fuenmayor and Benzmüller 2019b, §3.3). This assumption about goodness, however, seems to have been tacit in Gewirth. This is one way in which the use of interactive proof assistants makes explicit as an assumption what may seem to us unnecessary in an argument (whether because it is not noticed at all or because it seems entirely obvious).

Fuenmayor and Benzmüller (2019b, §4) give a full computationally-verified proof of Gewirth's principle that follows Gewirth's original reasoning; it takes 45 lines from 'proof' to 'QED'. This computer-checked formalization is evidence for the interpretation of Beyleveld (2012): Fuenmayor and Benzmüller have effectively shown that Beyleveld's reading of Gewirth's meta-ethical argument is valid, which is usually taken as supporting a textual reconstruction.

Fuenmayor and Benzmüller's reconstruction is also uploaded in the *Archive of Formal Proofs*—a repository for proofs encoded in *Isabelle*—just as the reconstruction of Aristotle's syllogistic by Koutsoukou-Argyraki is. Anyone doubtful about the steps in the proof can run the code on their own desktop version of *Isabelle*. However, as Fuenmayor and Benzmüller (2019b, §4) note (and as inspection of their proof shows), their proof in *Isabelle* is partly interactive and partly automated: some of the work is done by *Isabelle* under the hood, as it were. In some steps of this formalization, the user sets a goal for *Isabelle* and the theorem prover seeks a solution through, for instance, simplification (term rewriting). These steps are recoverable for human review, but they also spare us the human effort of discovering (often simple) steps. This is yet another benefit of leveraging interactive theorem provers (in particular, those with some automated theorem proving capabilities).

5 The future of interactive theorem provers in history of philosophy

The example of applying an interactive proof assistant to a text shows that interactive theorem provers can benefit those who deploy them in doing philosophical history. In the example of *Principia*, a complete reconstruction of the entire propositional logic was computer checked. It is not claimed that this is the only possible interpretation of the reasoning given in the text or that interactive theorem provers were indispensable for this task, but others who wish to criticize, or improve on, or just verify the reconstruction given are easily able to do so by downloading and editing the source files. Then the various reconstructions can be considered and evaluated on their merits and demerits much like interpretations made without interactive theorem provers are.

It is worth noting also that *Principia* is not exceptional in providing mere recipes rather than full step-by-step arguments. In many past philosophical works—say, Foot's "The Problem of Abortion" or Maimonides' *Guide for the Perplexed*—arguments are rarely given with the step-by-step completeness and rigor demanded by interactive theorem provers.[42] This is not to say that such works do (or do not) fall short of some high standard of rigor familiar to modern logicians. It is to say that, like *Principia*, many works mostly contain argument sketches: some steps are missing. Accordingly, formalization in interactive theorem provers can deepen our understanding the reasoning given in a text, especially by automating more tedious validity checking and copying tasks usually done manually.[43]

Precisely these benefits were enjoyed in formalizing *Principia*'s propositional logic proof recipes as full step-by-step proofs in the interactive theorem prover *Coq*. Further, although constructing full proofs in *Coq* often revealed omissions in the demonstration, there were often patterns to the omissions. The *Coq* formalization has induced a clearer picture of Whitehead and Russell's system for omitting theorems and of the complete step-by-step proofs indicated by *Principia*'s demonstrations.

This novel application of *Coq* to *Principia* shows the applicability of interactive proof assistants in textual reconstruction, which is of course interesting to anyone in one field (or more than one field) such that a crucial feature

[42] This is also true of much modern mathematical writing at the research level. Without going so far as (Gonthier 2005, 2) and (Hales et al. 2015, 3) do and claiming that superior logical safety belongs to *Coq*-checked proofs over the human-reviewed ones, we can nonetheless say that the formalization requirements for mathematical research today fall far short of the standards laid out in the QED Project (Boyer 1994, 238). Indeed, the late Vladimir Voevodsky's work on Univalent Foundations brought new focus and energy into the QED Project (Voevodsky 2015, 1278). So long as mathematicians today are publishing proof recipes (if they are), then there is room for such formalization work as Gonthier, Hales, and others have carried out on modern texts—work no different in kind than what we carried out on *Principia*.

[43] "It is suggested that the time is ripe for a new branch of applied logic which may be called "inferential" analysis, treating proofs as numerical analysis does calculations. This discipline, it is believed, will in the not too remote future lead to proofs of difficult theorems by machine." (Wang 1963, 224)

of texts to be reconstructed (perhaps the crucial feature) is the arguments it contains, and not only to researchers in computer science, philosophy, history of science, or mathematics.

Finally, although I focused on reconstructing a single text, there is no reason in principle why interactive theorem provers could not be applied to a corpus more broadly—the Aristotelian corpus comes to mind—in a way that allows for cross-textual comparison and computer verification that one's reconstruction of, say, *De Anima* does not conflict with one's reconstruction of *Physica*. Similarly, one could compare a stretch of Bradley's *Appearance and Reality* with a passage in Russell's *Principles of Mathematics* to computer check that one's reconstruction of what Russell's passage contends really does conflict with or contradict what is defended in Bradley's work. So one could leverage interactive theorem provers in dialectical studies, too. These are perhaps ambitious suggestions about the potential uses for interactive theorem provers, but there is no principled reason why such applications are impossible: rational reconstruction using interactive proof assistants might be applied in the future to a whole corpus or between philosophers or traditions, mirroring the broad scope of past rational reconstructions done without interactive theorem provers.

Setting aside speculation about the future developments in applying interactive theorem provers, I conclude by summarizing what has been shown. The example discussed above, of applying an interactive theorem prover in reconstructing the reasoning in *Principia*'s text, shows in detail the sort of benefits that result from applying interactive theorem provers in doing philosophical history and precisely how they can arise from being fruitfully constrained by computer verification.

The many examples cited in §1 show further that the benefits of computer verification for historians of philosophy are not either an isolated incident or an idle fancy: they have been and are actively being realized by those using them. This is not to say that they are indispensable—just that they are useful, in the same way that other applications like text editing ones are. The abundance of examples shows that computer verification can be good for historians of philosophy, at least for those who want it.

Acknowledgements I thank participants of the 2021 meeting of the Society for the Study of the History of Analytical Philosophy for their helpful comments. I also thank Katalin Bimbó, Bernard Linsky, and four anonymous reviewers with *Synthese* for their excellent feedback and criticisms. I am grateful to Samuel C. Fletcher for organizing this special issue of *Synthese* and for his patience with my revisions. Some of the research on this paper was done while I was an Izaak Walton Killam Postdoctoral Fellow in Philosophy at the University of Alberta.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Alama J, Oppenheimer PE, Zalta EN (2015) Automating Leibniz's theory of concepts. In: International Conference on Automated Deduction, Springer, pp 73–97

Awodey S, Pelayo A, Warren MA (2013) Voevedsky's Univalence Axiom in Homotopy Type Theory. Notices of the American Mathematical Society 60(9):1164

Barendregt H, Barendsen E (2002) Autarkic computations in formal proofs. Journal of Automated Reasoning 28(3):321–336

Beaney M (2013) Analytic Philosophy and History of Philosophy: The Development of the Idea of Rational Reconstruction. In: Reck EH (ed) The Historical Turn in Analytic Philosophy, chap 9, pp 231–260, DOI 10.1007/978-1-137-30487-2

Benzmüller C, Paleo BW (2015) Interacting with Modal Logics in the Coq Proof Assistant. In: Beklemishev LD, Musatov DV (eds) Computer Science – Theory and Applications, Springer, Switzerland, Lecture Notes in Computer Science, Vol. 9139, vol CSR 2015, pp 398–411, DOI 10.1007/978-3-319-20297-6

Benzmüller C, Claus M, Sultana N (2015) Systematic verification of the modal logic cube in Isabelle/HOL. arXiv preprint arXiv:150708717

Beyleveld D (1991) The dialectical necessity of morality. University of Chicago Press

Beyleveld D (2012) The Principle of Generic Consistency as the Supreme Principle of Human Rights 13:1–18, DOI 10.1007/s12142-011-0210-2

Blumson B (2021) Mereology. Archive of Formal Proofs `https://isa-afp.org/entries/Mereology.html`, Formal proof development

Boyer R (1994) The QED Manifesto. In: Bundy A (ed) Automated Deduction — Cade-12: 12th International Conference on Automated Deduction, Proceedings, Berlin Heidelberg, Lecture Notes in Artifical Intelligence, vol 12, pp 237–251, in the panel discussion "A Mechanically Proof-Checked Encyclopedia of Mathematics: Should We Build One? Can We?"

Castagnoli L (2010) Ancient Self-Refutation. Cambridge University Press

Davidson D (1973) Radical Interpretation Interpreted 8(3/4):313–328, URL `https://www.jstor.org/stable/42968535`

Davidson D (1994) Radical Interpretation Interpreted 8:121–128, URL `https://www.jstor.org/stable/2214166`

Elkind LDC (2021) Russell on the Ethical Value of Logic. In: Madigan TJ, Stone P (eds) Bertrand Russell: Public Intellectual, 2nd edn, Tiger Bark Press, chap 15, pp 251–267

Fitelson B, Zalta EN (2007) Steps toward a computational metaphysics. Journal of Philosophical Logic 36(2):227–247

Fleuriot J (2001) A Combination of Geometry Theorem Proving and Nonstandard Analysis with Application to Newton's *Principia*. Distinguished Dissertations, Springer-Verlag, DOI 10.1007/978-0-85729-329-9

Fuenmayor D, Benzmüller C (2017) Automating emendations of the ontological argument in intensional higher-order modal logic. In: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz), Springer, pp 114–127

Fuenmayor D, Benzmüller C (2018a) Computational hermeneutics: An integrated approach for the logical analysis of natural-language arguments. In: International Conference on Logic and Argumentation, Springer, pp 187–207

Fuenmayor D, Benzmüller C (2018b) Formalisation and evaluation of Alan Gewirth's proof for the principle of generic consistency in Isabelle/HOL URL `https://www.isa-afp.org/entries/GewirthPGCProof.html`

Fuenmayor D, Benzmüller C (2019a) A computational-hermeneutic approach for conceptual explicitation. In: Nepomuceno-Fernández Á, Magnani L, Salguero-Lamillar FJ, Barés-Gómez C, Fontaine M (eds) Model-Based Reasoning in Science and Technology, Springer International Publishing, pp 441–469

Fuenmayor D, Benzmüller C (2019b) Harnessing Higher-Order (Meta-)Logic to Represent and Reason with Complex Ethical Theories. In: Nayak A, Sharma A (eds) PRICAI 2019: Trends in Artificial Intelligence, Lecture Notes in Computer Science, vol 11670, Springer, pp 418–432, URL `https://doi.org/10.1007/978-3-030-29908-8_34`

Fuenmayor D, Benzmüller C (2020) A Case Study on Computational Hermeneutics: E. J. Lowe's Modal Ontological Argument, Springer International Publishing, pp 195–228. DOI 10.1007/978-3-030-43535-6_12, URL https://doi.org/10.1007/978-3-030-43535-6_12

Geuvers H (2009) Proof assistants: History, ideas and future 34:3–25, DOI 10.1007/s12046-009-0001-5, URL https://link.springer.com/article/10.1007/s12046-009-0001-5

Gewirth A (1981) Reason and Morality. University of Chicago Press

Gonthier G (2005) A computer-checked proof of the Four Colour Theorem. URL https://www.cl.cam.ac.uk/~lp15/Pages/4colproof.pdf

Gonthier G (2008) A Computer-Checked Proof of the Four Colour Theorem. Notices of the American Mathematical Society 55(11):1382–1393, URL http://www.ams.org/notices/200811/tx081101382p.pdf

Gonthier G, Asperti A, Avigad J, Bertot Y, Cohen C, Garillot F, Roux SL, Mahboubi A, O'Connor R, Ould SB, Pasca I, Rideau L, Solovyev A, Tassi E, Théry L (2013) A Machine-Checked Proof of the Odd Order Theorem. In: Blazy S, Paulin-Mohring C, Pichardie D (eds) Interactive Theorem Proving, Springer Berlin Heidelberg, Berlin, Heidelberg, Lecture Notes in Computer Science, pp 163–179, URL https://hal.inria.fr/hal-00816699

Hales T, Adams M, Bauer G, Dang DT, Harrison J, Hoang TL, Kaliszyk C, Magron V, McLaughlin S, Nguyen TT, Nguyen TQ, Nipkow T, Obua S, Pleso J, Rute J, Solovyev A, Ta AHT, Tran TN, Trieu DT, Urban J, Vu KK, Zumkeller R (2015) A Formal Proof of the Kepler Conjecture. ArXiv e-prints URL https://arxiv.org/abs/1501.02155, eprint arXiv:1501.02155

Harnad S (1990) Against computational hermeneutics 4:167–172, URL http://cogprints.org/1576/

Harrison J (1996) Proof style. In: International Workshop on Types for Proofs and Programs, Springer, pp 154–172

Harrison J, Urban J, Wiedijk F (2014) History of interactive theorem proving. In: Computational Logic, vol 9, pp 135–214

Hawthorne J (2021) Inductive Logic. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, spring 2021 edn, Metaphysics Research Lab, Stanford University

Hilbert D (1899) Grundlagen der Geometrie. Verlag von B. G. Teubner

Kirchner D, Benzmüller C, Zalta EN (2019) Computer science and metaphysics: A cross-fertilization. Open Philosophy 2(1):230–251

Kirchner D, Benzmüller C, Zalta EN (2020) Mechanizing Principia Logico-Metaphysica in Functional Type-Theory. The Review of Symbolic Logic 13(1):206-218, DOI 10.1017/S1755020319000297

Klein A (2020) Philosophy at War: Nationalism and Logical Analysis. URL https://aeon.co/essays/philosophy-at-war-nationalism-and-logical-analysis

Koutsoukou-Argyraki A (2019) Aristotle's assertoric syllogistic. Archive of Formal Proofs https://isa-afp.org/entries/Aristotles_Assertoric_Syllogistic.html, Formal proof development

Kremer M (2013) What Is the Good of Philosophical History? In: Reck EH (ed) The Historical Turn in Analytic Philosophy, chap 11, pp 294–325, DOI 10.1007/978-1-137-30487-2

Lapointe S, Pincock C (2017) Introduction. In: Lapointe S, Pincock C (eds) Innovations in the History of Analytical Philosophy, Palgrave Macmillan, pp 1–23

Lokhorst GJC (2010) Where Did Mally Go Wrong? In: Deontic Logic in Computer Science, Springer, vol 6181, URL https://doi.org/10.1007/978-3-642-14183-6_18

Lokhorst GJC (2011) Computational meta-ethics. Minds and machines 21(2):261–274

Maric F (2015) A survey of interactive theorem proving. Zbornik radova 18(26):173–223

McCandless D (2015) Codebases. URL https://www.informationisbeautiful.net/visualizations/million-lines-of-code/, research by Pearl Doughty-White and Miriam Quick

Mohr JW, Wagner-Pacifici R, Breiger RL (2015) Toward a computational hermeneutics 2(2):2053951715613809, DOI 10.1177/2053951715613809, URL https://doi.org/10.1177/2053951715613809, https://doi.org/10.1177/2053951715613809

Müller D, Rabe F, Sacerdoti Coen C (2019) The *Coq* library as a theory graph. In: Kaliszyk C, Brady E, Kohlhase A, Sacerdoti Coen C (eds) Intelligent Computer Mathematics, Springer International Publishing, Cham, pp 171–186

Novak N (2015) Practical Extraction of Evidence Terms from Common-Knowledge Reasoning. Electronic Notes in Theoretical Computer Science 312(24):143–160, DOI https://doi.org/10.1016/j.entcs.2015.04.009

O'Leary DJ (1988) The Propositional Logic of *Principia Mathematica* and Some of Its Forerunners. Russell: The Journal of Bertrand Russell Studies 8(1-2):92–115

Parsons T (2021) The Traditional Square of Opposition. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, Fall 2021 edn, Metaphysics Research Lab, Stanford University

Piotrowski M, Neuwirth M (2020) Prospects for Computational Hermeneutics. In: Atti del IX Convegno Annuale AIUCD, DOI 10.6092/UNIBO/AMSACTA/6316

Plato (1997) Sophist. In: Cooper JM (ed) Plato: Complete Works, Hackett Publishing Company, pp 235–293

Portoraro F (2019) Automated Reasoning. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, spring 2019 edn, Metaphysics Research Lab, Stanford University

Reed D (2005) Figures of Thought: Mathematics and Mathematical Texts. Routledge

Rockwell G, Sinclair S (2016) Hermeneutica: Computer-assisted interpretation in the humanities. MIT Press

Rorty R (1984) The Historiography of Philosophy: Four Genres. In: Rorty R, Schneewind JB, Skinner Q (eds) Philosophy in History, Cambridge University Press, chap 3, pp 49–76

Russell B (1914/1986) On Scientific Method in Philosophy. In: Slater JG (ed) The Philosophy of Logical Atomism and Other Essays, 1914-19, The Collected Papers of Bertrand Russell, vol 8, George Allen & Unwin LTD., New York, chap 5, pp 55–73

Saqib Nawaz M, Malik M, Li Y, Sun M, Ikram Ullah Lali M (2019) A Survey on Theorem Provers in Formal Methods. arXiv e-prints arXiv:1912.03028, `1912.03028`

Shapiro S, Kissel TK (2018) Classical Logic. Stanford Encyclopedia of Philosophy URL `http://plato.stanford.edu/archives/spr2018/entries/logic-classical/`

Smith R (2020) Aristotle's Logic. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, fall 2020 edn, Metaphysics Research Lab, Stanford University

Szabó ZG (2020) Compositionality. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, Fall 2020 edn, Metaphysics Research Lab, Stanford University

Voevodsky V (2015) An Experimental Library of Formal Mathematics Based on the Univalent Foundations. Mathematical Structures in Computer Science 25(5):1278–1294, DOI 10.1017/S0960129514000577

Wang H (1963) Toward Mechanical Mathematics. In: A Survey of Mathematical Logic, 1, Science Press and North-Holland Publishing Company, chap IX, pp 224–268, reprinted from 1960

Wenzel M (2021) The Isabelle/Isar Reference Manual. URL `https://isabelle.in.tum.de/documentation.html`

Whitehead AN, Russell B (1910) Principia Mathematica, vol I, 1st edn. Cambridge University Press

Williams B (1994) Descartes and the Historiography of Philosophy. In: Cottingham J (ed) Reason, Will, and Sensation: Studies in Descartes's Metaphysics, Clarendon Press