Minds & Machines (2007) 17:185–202
DOI 10.1007/s11023-007-9063-5

Computers, Justification, and Mathematical Knowledge

Konstantine Arkoudas · Selmer Bringsjord

Published online: 23 June 2007

Abstract The original proof of the four-color theorem by Appel and Haken sparked a controversy when Tymoczko used it to argue that the justification provided by unsurveyable proofs carried out by computers cannot be a priori. It also created a lingering impression to the effect that such proofs depend heavily for their soundness on large amounts of computation-intensive custom-built software. Contra Tymoczko, we argue that the justification provided by certain computerized mathematical proofs is not fundamentally different from that provided by surveyable proofs, and can be sensibly regarded as a priori. We also show that the aforementioned impression is mistaken because it fails to distinguish between proof search (the context of discovery) and proof checking (the context of justification). By using mechanized proof assistants capable of producing certificates that can be independently checked, it is possible to carry out complex proofs without the need to trust arbitrary custom-written code. We only need to trust one fixed, small, and simple piece of software: the proof checker. This is not only possible in principle, but is in fact becoming a viable methodology for performing complicated mathematical reasoning. This is evinced by a new proof of the four-color theorem that appeared in 2005, and which was developed and checked in its entirety by a mechanical proof system.

Keywords A priori · Justification · Proofs · Certificates · Four-color theorem · Mathematical knowledge

K. Arkoudas (✉) · S. Bringsjord
Cognitive Science Department, Computer Science Department, RPI, Troy, NY, USA 12180
e-mail: arkouk@rpi.edu

S. Bringsjord
e-mail: brings@rpi.edu

Introduction

A controversy started shortly after the appearance of the original ''computer proof'' of the four-color theorem by Appel and Haken (1977). The controversy was initiated by Tymoczko (1979), and was mostly played out on the pages of the *Journal of Philosophy*.[1] Tymoczko's article claimed that the work of Appel and Haken heralded a radically new conception of mathematical proof. The reason, in his view, was that the proof of Appel and Haken could not be surveyed in its entirety by human mathematicians. Consequently, he maintained, the proof constituted inherently *a posteriori* justification for the four-color theorem. He also regarded the proof as a piece of experimental mathematics, in the same sense that ''experimental'' is understood in the natural sciences, and as demonstrating that mathematical knowledge, far from being infallible, has a degree of certainty comparable to the results of empirical sciences.

We argue that the justification provided by unsurveyable computerized formal proofs is not epistemically different from the justification provided by surveyable formal proofs, as long as the algorithms that the computers execute are known to be correct in the traditional sense. Accordingly, computerized proofs can provide a priori mathematical justification if surveyable proofs can. We introduce a new notion of ''computational a priori'' knowledge that helps to clarify some of these issues. Because proof-checking algorithms are small and simple, a priori knowledge of their correctness seems possible, and therefore it might well be possible to know a priori a result established by a proof that was checked by such an algorithm, regardless of whether the algorithm was applied by a human or by a reliable machine. From a practical perspective, we can use computers to attain this type of knowledge if we draw a sharp distinction between discovery and justification, and specifically if we require discovery software to emit low-level correctness proofs—called ''certificates''—that can be verified by simple proof checkers. This methodology is becoming increasingly feasible. Although the original proof by Appel and Haken did not produce certificates that could be checked by small and simple proof checkers, this was accomplished in 2005 by Georges Gonthier, who formalized and completely checked the proof with the mechanical proof assistant Coq (Gonthier 2005).

Beliefs About Proof Correctness

Tymoczko (1979, p. 59) wrote that ''proofs are surveyable,'' but that is ambiguous, as it could be taken to mean that all proofs can be surveyed by humans, which is plainly false. If the concept of proof was limited only to proofs that could actually be surveyed, mathematics would cease to exist as we know it for the simple reason that there is only a finite number of such proofs, no more, say, than the number of atoms in the universe, call it n. But if there are only n theorems in mathematics, no matter how large n might be, the field degenerates into a finite list of sentences.

[1] See Detlefsen and Luker (1980) and Teller (1980) for some replies.

Theoremhood would become mechanically decidable, since, trivially, any finite set is decidable. In fact such a limitation would be inconsistent as long as we accepted that there are infinitely many natural numbers, for with each number i we can associate a true mathematical proposition (e.g., that i is even or odd, or that the ith digit in the decimal expansion of π is such-and-such), each of which has a straightforward demonstration. That there are infinitely many proofs is something so fundamental and deeply woven into the fabric of mathematics that to reject it would be to reject mathematics.[2] Accordingly, Tymoczko was wrong to state categorically that ''proofs are surveyable.'' What he should have said instead is that proofs are surveyable *in principle*, but not necessarily *in practice*. Only *some* proofs can actually be surveyed, just as only some natural numbers can actually be written down and manipulated, only some Turing machines can actually be built, and so on. But just as we do not let empirical limitations constrain our *concept* of natural numbers, or our concept of Turing machines, or our concept of circles and squares, or pretty much any other mathematical concept, so it is with proofs; we do not let the limitations of our brains and the physical universe in general dictate what constitutes a proof. Similar remarks apply to several other of Tymoczko's proclamations, e.g., his statement that ''A proof is a construction that can be looked over, reviewed, verified, by a rational agent'' (1979, p. 59). What he should have said is that a proof is an object that can be verified by an *idealized* rational agent, not by any rational agent. Tymoczko elsewhere acknowledges the obvious point that ''formal proofs outrun surveyable proofs,'' but he should have said simply that *proofs* outrun surveyable proofs, because there are infinitely many informal proofs as well.

Now consider a mathematical proposition all of whose proofs happen to be humanly unsurveyable in practice (in the actual world). Does it make sense to say that such a proposition is knowable a priori? We believe that a moderate rationalist could cogently argue for an affirmative answer, and in what follows we will briefly sketch a picture of what kind of sense can be made of such a claim.[3] But it should be stressed that such questions can easily get sidetracked into futile terminological disputes. Different philosophers have quite different understandings of aprioricity, after all. In Chisholm's conception, for instance, only propositions that are themselves evident or else follow from evident propositions via evident entailment are knowable a priori (Chisholm 1989). Roughly, a proposition is knowable a priori only if it can be apprehended as true by way of direct and immediate rational insight, to use the terminology of Bonjour (1998). Long deductions are substantively underwritten by memory and the physical underpinnings of various mental states, and therefore the demonstrative justification they provide cannot be priori. In such a view, even humanly surveyable proofs do not constitute a priori justifications for their conclusions if they are long or complicated. And Kitcher (1983) put forth

[2] Strict finitists and adherents of other extreme forms of constructivism might take issue with that, but we will subscribe to what has been the received view ever since Cantor and endorse the existence of infinitely many mathematical objects of various types, including proofs. To the extent that it is possible, we will try not to let this prejudge any epistemological questions, particularly in connection with aprioricity.

[3] Burge(1998) and others have expressed similar positions, and we owe a debt to them, but our analysis here will take a different route.

an analysis of aprioricity that requires infallibility and indefeasibility, so that if subsequent experiences could undermine what we previously held to be a justified belief, the justification could not have been a priori after all—a conception that was criticized as somewhat of a strawman, given that few apriorists today would consider mathematical knowledge to be either infallible or indefeasible.[4] Alternatively, one could argue that memory (or a computer, for that matter) is a physical mechanism that is crucial in causing us to believe that a proof demonstrates a certain proposition, but that the warrant provided by the proof itself is a priori. We will now explore these ideas further, but keeping in mind that there is no widely accepted analysis of aprioricity, and that the correct view of the matter might ultimately turn out to be deflationary.

Suppose a mathematician X checks a long and complicated formal proof D of a mathematical result P, and convinces himself that the proof is sound and hence that P is true. The verification was long and tedious, requiring a lot of paperwork and patience. On what grounds does X now say that he knows P? Is it on the grounds of his properly functioning memory, his good eyesight, his powers of attention and mental concentration, his patience and meticulousness? Suppose he does say that. Now Y and Z and other mathematicians also check the proof and convince themselves of its soundness. Y now also says that she knows P on the basis of *her* memory and eyesight and so on, and Z says that he knows P on the basis of *his* memory and eyesight, and so forth. But there now seems to be little in common among all these cases on the basis of which P is known. So a causally oriented epistemologist (e.g., a reliabilist) might instead say that the grounds on which X believes P consist in the *psychological process* that occurs when one reads, understands, and assents to a mathematical proof. But note that this is not what mathematicians themselves usually say. What they say is that P is known *on the grounds of the proof* D, not on psychological grounds. Empirical aspects of the psychological process of reading a mathematical proof of P do not enter into the epistemic warrant that justifies P. The psychological process in question is indeed essential in *causing* us to acquire the belief that D is a correct proof that establishes P. But we should not confuse the physical etiology of a mathematical belief with its epistemic warrant. When we are asked to justify our belief we do not point to any set of contingent facts; we point to a set of necessary facts. Of course, we might be mistaken. Any belief-fixation process, being physical, can lead us astray. But the warrant itself, when one is to be had, is a priori.

This is, in fact, the standard rationalist view of mathematical proofs, which takes them to provide a priori knowledge of the implication between premises and conclusion.[5] Even if we do not adopt this view, however, an argument can be made

[4] In ''a mixture of penitence and intransigence,'' Kitcher has more recently revised his earlier account (2000), but continues to maintain that mathematical knowledge cannot be a priori, and that any philosophically interesting account of aprioricity must entail infallibility and indefesability.

[5] In the present paper, when we speak of the proposition established by a proof we will mean the conditional proposition to the effect that the premises entail the conclusion. Whether or not the premises are known a priori is another question. We believe that quite a few mathematical truths can be known a priori by direct rational insight instead of inference (Peano's axioms, for example), even though some others (such as certain set-theoretic axioms) might not be so knowable.

that *if* we regard surveyable proofs as providing a priori justification, or at least justification that is not *essentially* dependent on experience, then we should also also regard the justification provided by certain kinds of computer-checked proofs as a priori in the same sense, because there are no significant epistemic differences between the two cases. This would undercut Tymoczko's argument, because even though he did allow for a priori mathematical knowledge on the basis of traditionally surveyable proofs (unlike, say, Kitcher, who thinks that there is no a priori knowledge in mathematics whatsoever), he nevertheless ruled out a priori justification issuing from unsurveyable computer-checked proofs. If our analysis is correct, this combination of positions is untenable. Some unsurveyable computer-checked proofs can provide a priori justification if any surveyable proofs can.

Let us return to the scenario of mathematician X who verifies that a formal proof D soundly leads to the conclusion that P. Suppose we ask X what justifies him in believing P. The dialogue would most likely proceed along the following lines:

Q_1: On what grounds do you claim to know P?
A_1: There is a sound proof for it, D, and I know that the conclusion of any sound proof is valid.
Q_2: And what are your grounds for believing that D is sound?
A_2: The fact that I inspected a copy of it and verified that it soundly produces the right conclusion.

Let us try to unpack these answers in greater detail. First it will help to distinguish between a formal proof D,[6] which we will view as an abstract type, and some concrete physical token of it, $\widehat{D}$. Proof tokens are physical instantiations of the corresponding abstract proofs; they can be written down on paper or other media, they can be read, inspected, erased, marked, copied, transmitted (potentially incorrectly), and so on. So X did not check a proof D per se, but rather some particular token $\widehat{D}$ of such a proof (this was reflected in X's comment about being given ''a copy'' of the proof). We will make a similar distinction between a formula F and a token $\widehat{F}$ of that formula. We will write F_P for a formula that expresses a proposition P.[7] We write $\models F$ to mean that the formula F is valid. We will not analyze validity in detail, but will assume that any proposition expressed by a valid formula is true. Further, implicit in X's foregoing remarks is that there is a proof-checking algorithm, which we will call PC, and which can be used to *check* any given proof D. We write P$C(D)\rightsquigarrow F$ to mean that applying the algorithm PC to D eventually produces the formula F, which indicates that D is sound, and that the proposition expressed by F is the conclusion that D establishes. We write P$C(D)\rightsquigarrow$*error* to indicate that the application of PC to D revealed an error, and that

[6] We assume that proofs are expressed in a sufficiently expressive formal system, and that there is a simple proof-checking algorithm that can be used to verify such proofs. That is the case, e.g., for first-order logic and for several higher-order type theories.

[7] We distinguish between formulas and propositions, the latter being what the former *express*, but nothing of what we say here hinges on the metaphysical status of propositions.

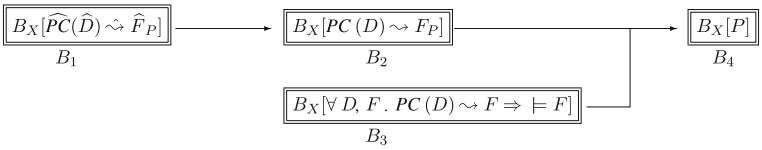

Fig. 1 Justificational structure of knowledge produced by surveying a proof

consequently D is unsound.[8] These are the only two output possibilities for any given D: either $PC(D) \rightsquigarrow F$ for some F which represents the conclusion of D; or else $PC(D) \rightsquigarrow error$.The proof-checking algorithm PC is also an abstract object, and when we write $PC(D) \rightsquigarrow F$, the designated relationship should be understood completely mathematically, without reference to any physical realizations of these objects. We will write $\widehat{PC}(\widehat{D}) \widehat{\rightsquigarrow} \widehat{F}$ to indicate that a putative token of the proof-checking algorithm (say, one that is stored in someone's brain, or one that is stored in a text file on a hard disk) is applied to a particular token of the proof and results in a particular token of formula F. Finally, we write $B_X[P]$ to mean that X believes P, for any proposition P. With this background, we can now graphically depict the justification process that was described in the foregoing dialogue as shown in Fig. 1.

The arrows indicate justificational support. The left-to-right ordering signifies temporal procession, culminating in the belief of X that P holds. This order is therefore the reverse of that in which the answers of X were given in the dialogue. The middle nodes, in particular, B_2 and B_3, correspond to answer A_1. Two beliefs of X were cited there: First, that ''there is a sound proof for P, D.'' This is analyzed as

[8] Some have expressed puzzlement as to how a proof can be unsound. In the words of Rota (1997, p. 183). ''The expression 'correct proof' is redundant. Mathematical proof does not admit degrees. A sequence of steps in an argument is either a proof, or else it is gibberish.'' However, the expression 'correct proof' is no more redundant than the expression 'correct computer program'. Both proofs and programs have rich, recursive syntactic structure (a fact that is obscured in the preceding passage, which suggests that proofs are linear sequences of steps). Both can be given inputs and evaluated, and the evaluation can generate an error, or it can generate an output (in the case of proofs, the output is the conclusion). Correctness is always relative to specification. A proof can perfectly correctly derive a conclusion C_1, but we might still call it incorrect if we expected it to derive some other conclusion C_2 instead. Moreover, even if it does derive the desired conclusion, the question is what assumptions it requires in order to do so. A proof could correctly generate a conclusion with respect to one set of assumptions and fail with respect to others. Sets of assumptions are the primary ''inputs'' to proofs. This view of proofs as complex syntactic objects with formal evaluation semantics over assumption bases is developed at length by Arkoudas (2000). In this paper, by a ''proof'' D we will generally mean a syntactic object, essentially an abstract syntax tree (AST) in some appropriate abstract grammar (Reynolds 1998). We will be quite liberal on what counts as a token $\widehat{D}$ of such a proof D; it could be either (a token of) a linear string over some alphabet that can be parsed into an AST, or (a token of) a two-dimensional picture of an AST, and so on. If a token $\widehat{D}$ is incorrectly mutated, then the resulting physical object might fail to be a token of any proof, or it might become a token of some other proof. Precise identity criteria for proofs can be given but need not concern us here.

$$PC(D) \rightsquigarrow F_P, \tag{1}$$

which is to say that applying the proof-checking algorithm to D to it yields a formula that expresses P. Second, that ''the conclusion of any sound proof is valid.'' This is in turn analyzed as

$$\forall D, F \,.\, PC(D) \rightsquigarrow F \Rightarrow \models F, \tag{2}$$

i.e., whenever the application of the proof-checking algorithm to a proof D yields a formula F, F is valid. From these two propositions it follows logically that F_P is valid, and therefore that P is true. X is aware of these implications, and is thus led to B_4, the belief in P.

The first node, B_1, corresponds to answer A_2, where X claims to have verified that ''a copy'' of D ''soundly produces the right conclusion.'' This is understood to mean that he has faithfully applied $\widehat{PC}$ to a token of that proof and obtained the result $\widehat{F}_P$. $\widehat{PC}$ should be regarded as a putative token of the proof-checking algorithm PC to which X attributes the pleasant property (2). It is presumably stored in X's brain. X believes that he really was following PC during his verification effort, which is to say that he believes that $\widehat{PC}$ is a faithful instantiation of an algorithm PC that satisfies (2), and thus he feels justified in moving from B_1 to B_2.

Strong apriorists might deny that belief B_1 provides *justification* for B_2. By their lights, the arrow from B_1 to B_2 indicates *causal* dependence only, not justificational dependence. There might well be a very strong case to be made for this view. We will not adopt this view here, however. Let us instead grant that the physical process of successfully applying $\widehat{PC}$ to input $\widehat{D}$ provides actual justification for B_2, so that, ultimately, the belief in P rests on empirical facts.[9] Even if we concede that some empirical facts need to be invoked in order to justify a belief in the correctness of a proof, this does not make the justification a posteriori in the customary sense of the term ''a posteriori.'' The dependence on experience that is at work here is of a very different sort from the dependence on experience that is typically found in the natural sciences. Whereas the a posteriori knowledge of the natural sciences has an essential dependence on empirical propositions, the dependence discussed here is not essential. We will later explain this in terms of the multiple realizability of algorithms. But for now we will sketch an intuitive outline of the sense in which the justification provided by a formal proof D is usually understood to be a priori.

We assume that an idealized mathematician, who has an understanding of P and knowledge of the proof-checking algorithm and its correctness, is equipped with:

1. Unbounded quantities of paper, part of which contains a token of the proof D, the rest being blank paper for auxiliary use.
2. Potentially infinite amounts of ink, time, and patience.

[9] Such facts will include both specific observations and generalizations, e.g., an assertion to the effect that humans can reliably carry out algorithms.

It is not necessary to assume that the mathematician is capable of intellectual feats impossible for mere mortals, or that he is infallible. Cognitively, he is an ordinary mathematician. He simply happens to have sufficiently large time and space resources, along with patience.[10] Given these, he can mechanically check the given proof and eventually, after an indefinitely long but finite period of time, *will* reach one of two verdicts: ''Yes, D is sound and P holds'' or ''No, D is erroneous, and thus P might not be true.'' It is in that sense that the warrant supplied by D is usually understood to be a priori. Our mathematician need never leave his armchair and need never make any empirical inquiries about the world in order to accomplish his task. His existing mental states along with his paper and ink form a closed epistemic system. No contingent empirical information about the outside world is necessary in order to acquire the knowledge that P,[11] no measuring devices, no microscopes, no field work, no experiments. This is in obvious contrast to scientific knowledge (in the natural sciences), whose hallmark method of acquisition is empirical testing rather than armchair calculation. To us this seems to mark a very clear distinction, regardless of whether we view it as epistemological or merely methodological. Under this understanding, every provable mathematical proposition is a priori knowable and the issue of computer assistance is tangential to the matter; the four-color theorem is knowable a priori, whether or not we actually enlist the help of a computer in verifying it.

The preceding idealization is consistent with the intuition that surveyability is not an essential epistemological property of proofs. Indeed, if we say that the justification afforded by a surveyable mathematical proof is a priori in that it has no essential dependence on experience or on facts about the world, it is very hard to see why a longer proof would be a posteriori. In what sense would the justificational powers of longer proofs be essentially dependent on empirical evidence? Of course one could simply deny that longer proofs even exist, but we think it obvious that doing so would be decidedly unmathematical. One might as well deny that large integers exist. Alternatively, one might admit that longer proofs exist, perhaps as meaningless strings in a free monoid do, but deny that they are capable of providing any justification, in effect equating justificational capacity with surveyability. But

[10] These are, of course, very similar to the idealizations made by Turing in his analysis of computation. An important difference is that Turing did not require any understanding beyond the ability to follow rules.

[11] There are some subtle issues here. If we require *direct understanding* of a proposition P in order for P to be known, then propositions expressed by very large formulas cannot be known at all—and a fortiori cannot be known a priori—by an ordinary person, even one who is equipped with unbounded time, paper, ink, and patience (although they could be known by persons with arbitrarily—but finitely—larger intellectual capacities than ours). Even instances of tautologies such as $x = x$ would be unknowable for sufficiently large values of x, such as

$$32957873202125534000048257362 = 32957873202125534000048257362, \tag{3}$$

since we cannot directly comprehend the proposition expressed by such a formula. However, ordinary persons can know a proposition indirectly, by description, as in ''the proposition expressed by (3)'' or ''the proposition established by the proof I just checked,'' provided we know that such definite descriptions designate unique propositions, and in that sense arbitrarily large proofs can indeed provide knowledge.

where would that surveyability line be drawn, after which proofs lose all epistemic value and become syntactic junk? At 10 pages of text? At 100 pages? At whatever length can be agreed upon by the mathematical community at some fixed point in time? It seems clear that any such limit would be ad hoc. The capacity of a proof to serve as an epistemic warrant is not something that derives from its size, but from its logical properties.

Computational A Priori Knowledge

The above characterization is an abstract specification of one prominent sense in which the justification provided by formal mathematical proofs is often understood to be essentially independent of empirical evidence. We will now go further by introducing a new notion of a priori knowledge that we call *computational a priori* knowledge, which will pertain to knowledge obtained not just from the application of proof-checking algorithms to proofs, but from the application of a much wider class of algorithms to a much wider class of inputs. According to this conception, and contra Tymoczko, the four-color theorem not only *can* but actually *is* known a priori at present (2007).[12] We nevertheless agree with Tymoczko that the proof of Appel and Haken (1977) did not result in a priori knowledge of the theorem, although, as we will explain, our reasons for that contention are very different from his.

There are some details that need to be handled with care, but the core idea is simple: Knowledge obtained by computations is a priori as long as we know what the relevant algorithms do and can prove mathematically—using traditional, surveyable proofs—that the algorithms produce correct results. If that is the case, then the justificational structure of our knowledge is similar to that which obtains for knowledge produced by surveyable proofs. We will thus argue that if one accepts that surveyable proofs result in a priori knowledge, then one should accept that knowledge obtained by implemented algorithms which are a priori known to be correct is also a priori. Indeed, the particular hardware platform on which an algorithm is implemented is entirely immaterial, as long as we have good reason to believe that the underlying physical mechanism is reliable. An algorithm can be carried out in silicon, in DNA, in human brains, using vacuum tubes, via electricity or via completely mechanical means (e.g., wheels and cards, as in Babbage's Analytical Engine), bottles and buckets, quantum mechanical phenomena, the population of China, toilet paper and stones (Weizenbaum 1976, p. 51) or an indefinite number of other extremely diverse physical systems ranging from macroscopic to microscopic. This property of algorithms is commonly known as *multiple realizability*, and has been the linchpin of functionalism in the philosophy

[12] Assuming that the usual axioms about the real numbers and some other equally evident truths are known a priori. In this paper we are not concerned with non-inferential a priori knowledge, so we will not try to establish that the premises of Gonthier's proof of the four-color theorem are known a priori. We emphasize, nevertheless, that our position is not simple if-thenism. As mentioned earlier, we do believe that *some* mathematical truths can be known a priori without inference.

of mind for decades.[13] We believe that Tymoczko did not fully recognize the importance of the distinction between software and hardware, and consequently failed to see that the results we obtain by actual computations in the real world have no *essential* dependence on any particular hardware platform, as long as we have a priori knowledge of the correctness of the underlying algorithms—algorithms being, after all, thanks to Turing's work, mathematical objects amenable to rigorous analysis and proof. Whenever we come to know a proposition by virtue of a concrete computation of a particular implementation of an algorithm, there are counterfactual situations in which we could have learned the same result by a computation of an entirely different implementation of that algorithm, underwritten by altogether different physical laws.

The notion of a *physical implementation* or *realization* of an algorithm plays a key role in our analysis. We will not provide a detailed account of that notion (for an attempt at such an account see, e.g., Chalmers (1996), Sect. 9.2)[14]), but the basic idea is simple, and is based on the work of theoretical computer scientists over the last forty years who studied what it means for one formal computational process to implement another. Essentially, the computational structure of one process must mirror the structure of the other, which is formally captured by requiring that there is an appropriate homomorphism mapping any computation—sequence of states—of one system to one of the other. The same idea is used for a physical system implementing a formal one, only now it is the causal structure of the physical system that must mirror the computational structure of the formal system, and the morphism must relate physical states to formal states. In addition, the implementation must be *lawful*, i.e., the morphism must obtain counterfactually for different inputs.

Further, we will say that an algorithm is *statement-generating* if all of its normal outputs[15] are formulas of some recursively enumerable logical system[16] having a fixed formal syntax and semantics, with the latter picking out a certain subclass of formulas as *valid*. We do not impose any restrictions on the form of the inputs. We assume that once the logic is interpreted, every formula expresses a unique proposition that is either true or false, and that valid formulas are true in every interpretation. We will say that a statement-generating algorithm A is *correct* if no terminating computation of it produces a formula that is not valid.

Finally, we will say that an agent X has *computational a priori* knowledge of a proposition P at time t iff there is a statement-generating algorithm A and a physical system M such that:

1. X knows a priori that A is correct.
2. X knows that M implements A.

[13] Of course one need not subscribe to functionalism (neither author does) in order to recognize the multiple realizability of algorithms.

[14] However, see Rapaport (1999) for an alternative account of implementation.

[15] That is, apart from error messages.

[16] This simply means that there is an algorithm for listing all and only the theorems of the system. We do not wish to consider, for instance, systems with uncomputable inference rules, and the recursive enumerability proviso blocks such systems.

3. *X* knows that a computation of *M* was carried out prior to *t* and resulted in a statement expressing *P*.

We take it as a given that if the first condition does not obtain then *X*'s knowledge of *P* could not be said to be a priori in any meaningful sense. For instance, if *X* knows of the correctness of *A* because he has read about it in a journal or heard about it from a reliable friend, then he might have knowledge of it but it is not a priori,[17] and in view of the last two conditions, that would make his knowledge of *P* a posteriori. Note that the type of a priori knowledge required by this condition is not necessarily computational a priori (that would make the definition circular), but it is important to point out that it could be, as long as as the recursion eventually bottoms out in knowledge of some algorithm's correctness that is non-computational a priori. This allows us to bootstrap the epistemic process and obtain a priori mathematical knowledge justified by extremely complicated proofs (humanly unsurveyable) by anchoring our warrants on very simple algorithms whose proofs of correctness *are* humanly surveyable.

We are using the term ''algorithm'' liberally, including both algorithms in the traditional pretheoretical sense, as in a high-level informal description of, say, Euclid's procedure, using a mixture of natural language and mathematical terminology; as well as programs in formal languages such as variants of the λ-calculus. Proofs of correctness are possible for both, only in the former case the proofs are informal while in the latter case they can be either informal or formal. That rigorous, surveyable, and convincing proofs of correctness are possible for *some* algorithms is not controversial; it has been established by the work of theoretical computer scientists and logicians such as Hoare, Dijkstra, Manna, and Scott.

The knowledge required by the two last conditions is a posteriori. It can be had with different degrees of certainty. For instance, if I have myself keyed in a token of the program to which the first condition refers, have repeatedly checked to make sure that the program has been correctly entered, have tested its behavior in sample runs and found it to conform to its abstract specification, and so on, then I will probably have higher confidence in the second claim than if I receive such testimony by someone else.

Let us now consider the claim that the type of knowledge singled out by this analysis is a species of a priori knowledge. Suppose that *X* does indeed satisfy all three conditions for some proposition *P* at some time point *t*, and we ask him for his reasons for believing *P*. The dialogue would most likely go as follows:

Q_1: On what grounds do you claim to know *P*?
A_1: I know *P* because *A* gives F_P as an output, and I know a priori that *A* is correct. So F_P is valid, and *P* is true.
Q_2: And what are your grounds for believing that *A* gives F_P as output?

[17] Some philosophers (Williams 1972) have denied that mathematical knowledge can ever result from testimony. Most hold that reliable testimony produces a posteriori knowledge. Burge (1993) has tendentiously argued that not only can one come to know a mathematical proposition via someone else's testimony, but can come to do so a priori. Burge's arguments are subtle and deserve careful attention, but his position is prima facie implausible.

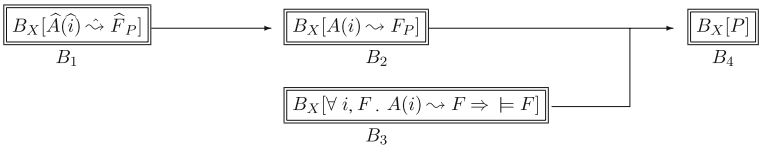

Fig. 2 Justificational structure of knowledge produced by a computerized proof

> A_2: The fact that A is implemented in this machine, and I know that a computation of that implementation produced F_P.

Accordingly, the process that resulted in X's belief that P can be graphically depicted as shown in Fig. 2, where, as before, we write $\widehat{A}$ to indicate a token of algorithm A. The middle beliefs, B_2 and B_3, represent the content of the first answer A_1. In particular, B_2 corresponds to X's claim that ''A gives F_P as an output'' (for a certain input), while B_3 corresponds to the claim that ''A is correct.'' The propositional contents of these beliefs *logically entail* P. X is aware of this implication and is thus led to B_4. The first node, B_1, corresponds to answer A_2, where X claims to know that a certain machine ''implements A'' and that ''a computation of that implementation produced F_P.'' The content of these beliefs is analyzed as $\widehat{A}(\widehat{i})\widehat{\rightsquigarrow}\widehat{F}_P$.

We thus see that the structure of the epistemic situation here is virtually identical to that of knowledge that results from a surveyable proof, as depicted by the diagram in Fig. 1. In this case, too, X feels justified in believing P by virtue of the existence of an appropriate mathematical object that serves as input to a computation of A, and more importantly, because he knows a priori that A is correct, i.e., that all terminating computations of it yield valid formulas. Let us compare the contents of the first empirical beliefs in the two cases. They are:

$$\widehat{PC}(\widehat{D})\widehat{\rightsquigarrow}\widehat{F}_P \tag{4}$$

in one case and

$$\widehat{A}(\widehat{i})\widehat{\rightsquigarrow}\widehat{F}_P \tag{5}$$

in the other. In the first case, that is, X believes that the instance of the proof-checking algorithm PC that he had in his mind/brain and which he applied to a certain proof token correctly yielded the result $\widehat{F}_P$. In the second case, X believes that the instance of the algorithm A that was realized in the machine at hand correctly produced the result $\widehat{F}_P$. The only difference lies in the first-person perspective that is present in one case but not in the other. Otherwise, both beliefs are based on physical experience and on empirical considerations. And given that certain computers are *much* more reliable than human brains, it would be eminently rational for X to believe (5) with higher confidence than (4), particularly if the proof token $\widehat{D}$ that he surveyed was large.

Consider the following passage from Tymoczko (1979, p. 72):

> [Belief in] formal [objects] cannot be used to legitimize the appeal to computers. Rather, we believe that the formal [object] exists only because we

> accept the appeal to computers in the first place. It is important to get the order of justification correct. Some people might be tempted to accept the appeal to computers on the ground that it involves a harmless extension of human powers. On their view the computer merely traces out the steps of a complicated formal proof that is really out there. In fact, our only evidence for the existence of the formal proof presupposes the reliability of computers.

That is surely right, even if Tymoczko's complaint is somewhat out of place because no one tried ''to legitimize the appeal to computers'' by citing any beliefs in the existence of the formal proofs which the computers presumably verify. The use of computers is legitimate simply and only because computers are reliable and, more importantly, because we know what they have been programmed to do. Beyond that, it is important to realize that what Tymoczko says above applies just as well to traditional surveyable proofs that do not involve computers, a fact that is made plain if we look at the leftmost justificational arrow of the diagram in Fig. 1. Our ''only evidence'' for the existence of a traditional surveyable formal proof D ''presupposes the reliability'' of our senses and our general cognitive abilities—yet Tymoczko did not argue on those grounds that the conclusion of such a proof is known a posteriori.

A second set of considerations suggesting that the type of knowledge specified by our analysis is a species of a priori knowledge falls out of the multiple realizability of algorithms. X could have had computational knowledge of P at t in a way that is not essentially dependent on M. In particular, A could have been implemented by a physical mechanism M' entirely different from M. So although X's knowledge needs *some* physical realization of A, it is not essentially dependent on any particular realization. To see this, consider two entirely different physical systems M and M', both implementing A, and underwritten by different sets of empirical laws, E and E'. Suppose that X uses both M and M' to perform the same computation, and obtains the same answer in both cases, some formula expressing P. On which set of empirical regularities is the justification of X's belief in P essentially dependent, on E or on E'? The two have nothing in common, so to say that it *essentially* depends on both would be gerrymandering (although it would be sensible to say that both computations contributed to X's justification). And it would be arbitrary to say that it essentially depends on either E or E' but not on the other. There is no essential dependence on either. The essential evidential factor is the algorithm that they both executed—the abstract object which both physical systems implemented.

We believe that such considerations show that what we have called computational a priori knowledge does not essentially depend on empirical factors, at least no more than surveyable proofs do. Nevertheless, whether we call one type of knowledge a priori and the other a posteriori is ultimately of little practical interest. The distinction itself, although genuine in our opinion, seems to us to be of secondary importance when it comes to computerized proofs. It might have been very important to Tymoczko, and it might still be of importance to other social constructivists, mainly, we think, because they often confuse aprioricity and infallibility, and perhaps because they have a postmodern ax to grind against any type of knowledge that might come across as ensconced in a lofty epistemic throne.

But what we think of as more important is the confidence with which we accept a proposition. Here there are no obscure distinctions between empirical and non-empirical sources of justification, only the brute fact of a simple partial order: We are more confident in our acceptance of some propositions than in others. And what is particularly interesting is the question of whether we can maximize these degrees of confidence for certain types of knowledge, and if so, how. It is in that light, we believe, that the subject of computer-assisted proofs should be considered—as a matter that calls for good engineering backed by solid metamathematics. And it is that perspective to which we will now turn our attention.

Maximizing our Confidence in the Correctness of Proofs

Our starting observation is that the Appel and Haken proof of the four-color theorem, by its very nature (heavily computational), in tandem with the controversy started by Tymoczko's paper, gave rise to a persistent misconception about computer proofs. It is this:

The fox view Computer proofs of non-trivial results, such as the four-color theorem, invariably depend on large amounts of *computation* performed by complicated custom-written software.

This view portrays the computer as a fox: It knows many different things in the sense that it executes an arsenal of various clever algorithms implemented for the explicit purpose of obtaining the result in question. This immediately raises very significant issues of trust. Even if we assume that all these algorithms are correct, how can we be sure that they are correctly implemented? Computer programs are notoriously prone to bugs. How can we trust page after page of C code written by the authors for the express purpose of verifying their results? This heavily computational view also conjures up a strong impression of experimentalism: We write some code, start executing it, and see what happens. Later on we might run the code again on a different machine and compare results; and so on. It *sounds* very experimental and empirical.

One result of this misconception was instilling an undue and non-discriminating distrust of computers in mathematics, as well as a blurring of the distinction between genuinely experimental mathematics (where only inductive evidence is provided for mathematical propositions) and computer-verified deductive proofs. For instance, more that a dozen years after the appearance of the Appel and Haken proof, we have statements such as the following one by Cohen (1991, p. 328)

> [a]dmitting the computer shenanigans of Appel and Haken to the the ranks of mathematics would only leave us intellectually unfulfilled.[18]

The misconception arises from a confusion between proof *search* and proof *checking*, or, in more traditional philosophical terms, between the *context of discovery* and the *context of justification* (Reichenbach 1938). A computer-conducted search for a deductive proof (in a fixed logical framework such as a

[18] Quoted by MacKenzie (1999, pp. 46–47), who provides an interesting sociological discussion of the four-color theorem.

Gentzen calculus or a Fitch-style system for classical first-order logic) can indeed carry out arbitrary amounts of computation, performed by arbitrary code. It can match patterns, unify terms, iterate without bounds, perform huge case analyses, etc. Indeed, if the underlying logic is interesting then the search might not even terminate. However, if and when a proof is actually found, checking that proof is trivial, and more importantly, it is not done by an arbitrary process. There is *one* fixed piece of code that can check *any* given proof. We use the term *certificate* to refer to a low-level formal proof produced by an arbitrarily complicated proof search.

If an arbitrarily complicated proof search produces a certificate, then, for justification purposes, the search has rendered itself obsolete. We no longer care how the search was performed, in how many pages of C code it was encoded, or what resources it used; our only care is whether the certificate is valid. This entitles us to think of the computer as a hedgehog, in that *it only needs to know one thing*, namely, how to check certificates in some appropriately rich logical framework (say, first-order predicate calculus with equality). To put it roughly, the computer only needs to know how to check applications of modus ponens and a small number of other trivial rules such as equality reflexivity and transitivity, and how to decide whether two formulas are identical. In terms of trust and justification, the hedgehog view implies that we only need to have faith in one algorithm: the proof-checking algorithm. This algorithm is of trivial conceptual and computational complexity. It can be implemented in a page or two of an advanced programming language such as ML or Haskell, and has linear-time average-case complexity in the size of the proof. In addition, it is completely domain-independent; the proof being checked could be about graphs or about polynomial roots or about metric spaces or about anything else whatsoever (e.g., an arbitrary proof in ZFC).

It is true that we still need to trust other things. To begin with, apart from the proof-checking algorithm, we need to believe that if a proof in the logic at hand exists, then the corresponding proposition holds. That is, we need to believe that the proofs are properly connected to the semantics.[19] We will take such fundamental theoretical beliefs as the lowest common denominator, and will assume that there is sufficient justification for them. Above that level, and more concretely, we need to trust the following components:

1. The various algorithms used by the underlying operating system (e.g., algorithms for implementing file operations on disk in terms of inode schemes).
2. The implementation of the underlying operating system (even if the aforementioned algorithms are mathematically correct, their implementation might have bugs).
3. The semantics of the programming language L in which the checker is implemented (e.g., does type soundness hold in L?).
4. The implementation of the compiler for L (bugs might creep into the compiler even if the semantics are theoretically sound).

[19] Of course we also need to believe that the semantics are right, that the system is consistent, and so on. But the deeper we go the more conceptual the issues become, and the less they have to do with the role of computers.

5. The integrity of the particular machine—hardware, operating system, and compiler for *L*—in which the checking is performed (e.g., we need to know that no malicious agents have tampered with the system).
6. General causal factors—random hardware malfunctions, influence of cosmic rays, and so on.

But we can do even better. We can eliminate the first four factors completely by implementing the proof-checking algorithm directly in silicon. That is, we can build special-purpose hardware whose only function is to check formal proofs in a particular logical framework. The design of that hardware can then be thoroughly—and easily—verified mathematically.[20] The only remaining possible seeds of doubt would then be of the fifth and sixth categories: hardware or software compromise by malicious intervention, and general ''physical stuff'' that could go wrong. The first can be dealt with by enforcing vigilant security measures. The second is inevitable, but it is also exceedingly unlikely. In any event, it is optimal, modulo the current state-of-the-art in hardware. We cannot do any better in this world. The possibility of error would then be far smaller than the possibility of error even in moderately complicated proofs that have been surveyed by humans.

One could object that the hedgehog view might be plausible in principle, but in practice, computer proofs such as the one by Appel and Haken do *not* output certificates that can be independently checked. We reply that this is a matter of engineering. When the Appel and Haken proof was developed in the 1970s, the required proof-engineering technology was not sufficiently mature.[21] But at present there are several proof systems that combine powerful proof search with proof checking, and are capable of producing certificates. For instance, many state-of-the-art resolution-based automated theorem provers—such as Vampire (Voronkov 1995) or Spass (Weidenbach 2001)—output linear deductions in the resolution calculus that can be independently checked by a very small program. Coq (Coquand and Huet 1998) is a proof system based on higher-order intuitionist type theory that also combines proof search with proof checking, and outputs certificates in the form of λ-calculus terms representing formal low-level deductions that can be checked independently.

The fact that Gonthier was able to use Coq to prove the four-color theorem constitutes solid evidence for the feasibility of the hedgehog view; sophisticated systems now exist that can carry out very demanding mathematical proofs and justify them using an extremely small domain-independent trusted base. (The original proof of Gonthier was developed in a slightly older version of Coq that is incompatible with the independent Coq proof checker that was recently developed at the University of Bologna, but Gonthier has recently finished porting his proof to

[20] Hardware verification can be challenging, so some have wondered whether we are substituting one difficult problem for another. But in the scenario we are envisaging we would not be concerned with the verification of *arbitrary hardware*. We would only be concerned with the verification of a specific, fixed hardware device, the one implementing the proof-checking algorithm. Moreover, since the specification of that device is so simple (assuming that it is mostly trust and simplicity that we are after, rather than efficiency), many of the complications that often arise in hardware verification—such as cache coherence protocol correctness—would not be an issue at all.

[21] The ideas were already there, but implementations were not powerful enough.

the new version, which will enable the independent proof checking.[22]) Another such system is our own Athena (Arkoudas n.d), which was designed and implemented with a view to minimizing the overall trusted base. Athena enables both proof search and proof checking in a Fitch-style system for polymorphic multi-sorted first-order logic, which, apart from making proofs more readable and writable, facilitates proof automation significantly. For technical reasons, certificates in Athena tend to be more compact than those of Coq, and can be optimized heavily using aggressive proof-simplification algorithms that do not perform cut elimination (Arkoudas 2005). But regardless of what system one chooses to use, the point remains that when it comes to trusting computerized proofs, we do not need to think of computers as foxes. They are hedgehogs. This is an important message that is well-known to researchers in the field of mechanized proof (Slaney 1994), but has yet to be conveyed to the wider mathematical and philosophical communities with sufficient clarity.

Acknowledgments We are grateful to Jim Fahey for insightful comments on an earlier draft of this paper. We also thank Amnon Eden and Ray Turner for organizing the PCS track of ECAP 2006 (at Norway's NTNU), and for comments they made on this paper during the conference.

References

Appel, K., & Haken, W. (1977). Every planar map is 4-colorable. *Illinois Journal of Mathamatics, 21*, 429–567.

Arkoudas, K. (2000). *Denotational proof languages*, PhD thesis, MIT, Department of Computer Science, Cambridge, USA.

Arkoudas, K. (2005). Simplifying proofs in Fitch-style natural deduction systems. *Journal of Automated Reasoning, 34*(3), 239–294.

Arkoudas K: n.d., Athena. http://www.pac.csail.mit.edu/athena

Bonjour, L. (1998). *In defense of pure reason*, Cambridge University Press.

Burge, T. (1993). Content preservation. *Philosophical Review, 102*, 457–488.

Burge, T. (1998). Computer proof, apriori knowledge, and other minds. *Noûs, 32*, 1–37.

Chalmers, D. J. (1996). *The conscious mind*. Oxford University Press

Chisholm, R. (1989). *Theory of knowledge* (3rd Ed.). Prentice-Hall.

Cohen, D. I. A. (1991). The superfluous paradigm. In J. H. Johnson & M. J. Loomes (Eds.), *The mathematical revolution inspired by computing* (pp. 323–329). Oxford: Clarendon Press.

Coquand, T., & Huet, G. (1988). The calculus of constructions. *Information and Computation, 76*, 95–120.

Detlefsen, M., & Luker, M. (1980). The four-color theorem and mathematical proof. *The Journal of Philosophy, 77*(12), 803–820.

Gonthier, G. (2005). A computer-checked proof of the four colour theorem http://www.research.microsoft.com/~gonthier/4colproof.pdf

Kitcher, P. (1983). The Nature of mathematical knowledge. Oxford University Press.

Kitcher, P. (2000). A priori knowledge revisited. In P. Boghossian & C. Peacocke (Eds.), *New essays on the a priori*. Oxford: Clarendon Press.

MacKenzie, D. (1999). Slaying the kraken: The sociohistory of a mathematical proof. *Social Studies of Science, 29*(1), 7–60.

Rapaport, W. J. (1999). Implementation is semantic interpretation. *The Monist, 82*, 109–130.

Reichenbach, H. (1938). Experience and prediction. University of Chicago Press.

Reynolds, J. C. (1998). Theories of programming languages. Cambridge University Press.

[22] Personal communication, January 22, 2006.

Rota, G. -C. (1997). The phenomenology of mathematical proof. *Synthese, 111*, 183–196
Slaney, J. (1994). The crisis in finite mathematics: Automated reasoning as cause and cure. In A. Bundy (Eds.), *Automated Deduction-CADE-12* (pp. 1–13). Springer: Berlin, Heidelberg.
Teller, P. (1980). Computer proof. *The Journal of Philosophy, 77*(12), 797–803.
Tymoczko, T. (1979). The four color theorem and its philosophical significance. *The Journal of Philosophy, 76*(2), 57–83.
Voronkov, A. (1995). The anatomy of Vampire: Implementing bottom-up procedures with code trees. *Journal of Automated Reasoning 15*(2).
Weidenbach, C. (2001). Combining superposition, sorts, and splitting, In: A. Robinson & A. Voronkov (Eds.), *Handbook of automated reasoning*, (Vol. 2). North-Holland.
Weizenbaum, J. (1976). *Computer power and human reason*, Freeman and Company
Williams, B. A. O. (1972). Knowledge and reasons. In G. H. V. Wright (Ed.), *Problems in the theory of knowledge* (pp. 1–11). Springer.