

Re: Article in Scientific American

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-12/msg06791.html>

- *From:* "T.H. Ray" <thray123@xxxxxxx>
 - *Date:* Tue, 26 Dec 2006 03:08:33 EST
-

T.H. Ray <thray123@xxxxxxx> wrote:

I am not quite sold. Montgomery-Smith's context was

that of computers checking computers.

I am not aware of such a context. The context was a discussion of whether a computer program should be considered to constitute a proof, in the sense that the program is run and if it prints "true", then that is considered sufficient evidence that the theorem is true without requiring additional proof.

Yes, but the probability of the computer program being correct does increase it if it is run on different computers with different compilers, operating systems, etc.<<

That sure sounds like not not provable to me.

Can you clarify what you mean by that? If the computer gives a proof, then the statement is provable. However, if the computer simply prints

out "true", then how does that "not not prove" the theorem?

The computer proof of a theorem actually rests on assumptions that not only is the theorem provable, but that it is not not provable. I.e., without an algorithm to verify a proof, we hand check the proof for logical gaps. Because an algorithm may contain no logical gaps, however, how does one know that the proof that a proof-checker produces is not merely proof of the validity of the proof-checker's algorithm?

(This case differs from the case in which computer aid is enlisted to calculate, or to recognize. The Appel-Haken proof of the 4C Theorem, e.g., assumed the validity of the computer proof only to the extent that the computer algorithm was able to recognize the identified finite set of maps. If proofs consisted only of counting and classifying, mathematics would be botany.)

The rest of your post is unintelligible to me. I don't mean that as a casual dismissal; I did try to read and understand it; it just appears to be a lot of generally true statements (with the exception of the phrase "or among computer proofs", which seems to beg the question) that don't pertain to this discussion.

Oh, yes, they do pertain in a most important way. You can't simply ignore Montgomery-Smith's point that agreement among computer methods increases the probability that a specific computer proof is true.

Again, the question was whether the existence of a computer program that outputs "true" should be considered a proof. I argued that it should not. My reason is that unlike in the case of a proof, the bulk of the program code is not read by other mathematicians as a primary means of understanding why the statement is true. Aside from being a significant difference in practical utility in itself, this means that errors in the

program are likely to be found only if they give verifiably incorrect output. That is equivalent to the theorem being disprovable. We've lost the potential that someone just realizes a gap in the reasoning even though the theorem is still true; or that the theorem is not disprovable but also not provable, so the program "proves" it in error, but no one will ever notice that fact.

Hence, relying on observed fact, that a computer program prints some output, as a proof amounts to subjecting the theorem to a less strong standard of truth, over sufficiently large spans of time.

[I'll again point out that if you write a computer program that produces a proof, then I have no problem with that proof, so far as it really does prove the proposition. Furthermore, if you (a) write a computer program, then (b) prove that it prints "true" only if the theorem is true, then (c) prove that it prints "true"; then I have no problem with the proof. It's only when step (c) is verified by empirical observation rather than proven that the "proof" is questionable, in my view... though in either case, there's probably a more straight-forward way to write the proof.]

—

Chris Smith

The question of whether a computer verification, of a computer proof, proves more than the validity of the algorithm that produces the answer, is significant.

Many important propositions are proved today as consequences of deeper theorems. FLT, e.g., as the consequence of the behavior of certain modular elliptic curves; the Poincare Conjecture as a result of a more general topology. Can proof programs checked by other

Re: Article in Scientific American

programs produce theorems whose proof implies the proof of the proof being checked?

In fact, you will find that I agree with your position, that a computer as theorem prover leaves a lot to be skeptical about. It is the potential for the computer to become a theorem producer that forces me to think in terms of not not provable. Why? Because heretofore we have known no way to differentiate "provable" from "not not provable." They are logically equivalent -- to us. To a calculating machine, fed input which allows no gap of logic, double negation is a necessary control against the bias of the programmer. And the computer's control against the bias of its program is another program.

One reaches the point of doing not ... mathematics ... but metamathematics. That's where Chaitin is leading, and that's exactly what he calls it. Should I be seduced?

Tom

.