What in the Name of Euclid Is Going On Here?

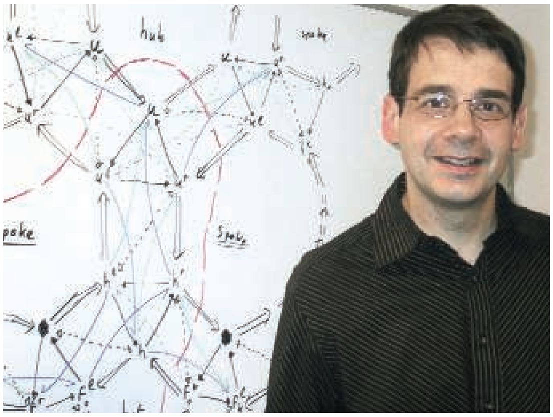
Computer assistants may help mathematicians dot the i's and cross the t's of proofs so complex that they defy human comprehension

In 1998, a young University of Michigan mathematician named Thomas Hales solved a nearly 4-century-old problem called the Kepler conjecture. The task was to prove that the standard grocery-store arrangement of oranges is, in fact, the densest way to pack spheres together. The editor of *Annals of Mathematics*, one of the most prestigious journals in mathematics, invited him to submit his proof to *Annals*. Neither of them was prepared for what happened next.

Over a period of 4 years, a team of 12 referees wrestled with the lengthy paper and eventually raised a white flag. They informed the editor that they were only "99 percent" certain that it was correct. In particular, they could not vouch for the validity of the lengthy computer calculations that were essential to Hales's proof. The editor took the unprecedented step of publishing the article with a disclaimer that it could not be absolutely verified (*Science*, 7 March 2003, p. 1513).

It is a scenario that has repeated itself, with variations, several times in recent years: A high-profile problem is solved with an extraordinarily long and difficult megaproof, sometimes relying heavily on computer calculation and often leaving a miasma of doubt behind it. In 1976, the Four Color Theorem started the trend, with a proof based on computer calculations so lengthy that no human could hope to follow them. The classification of finite simple groups, a 10,000-page multiauthor project, was completed (sort of) in 1980 but had to be recompleted last year. "We've arrived at a strange place in mathematics," says David Goldschmidt of the Institute for Defense Analyses in Alexandria, Virginia, one of the collaborators on the finite simple group proof. "When is a proof really a proof? There's no absolute standard." Goldschmidt thinks the traditional criterion—review by a referee (or team of them)—breaks down when a paper reaches hundreds or thousands of pages.

The computer—which at first sight seems to be part of the problem—may also be the solution. In the past few months, software packages called "proof assistants," which go through every step of a carefully written argument and check that it follows from the axioms of mathematics, have served notice that they are no longer toys. Last fall, Jeremy Avigad, a professor of philosophy at Carnegie Mellon University, used a computer assistant called Isabelle to verify the Prime Number Theorem, which (roughly speaking) describes the probability that a randomly chosen number in any interval is prime. And in December, Georges Gonthier, a computer scientist at Microsoft Research Cambridge, announced a successful verification of the proof of the Four Color Theorem, using a proof assistant called Coq. "It's finally getting to the stage where you can do serious things with these programs," says Avigad.



Mapping the way. Georges Gonthier's computer verified billions of calculations on "hypermaps" like the one shown.

CREDIT: COURTESY OF GEORGES GONTHIER

Even Hales is getting into the action. Over the past 2 years, he has taught himself to use an assistant called HOL Light. In January, he became the first person to complete a computer verification of the Jordan Curve Theorem, first published in 1905, which says that any closed curve drawn in the plane without crossing itself separates the plane into two pieces.

For Hales, the motivation is obvious: He hopes, eventually, to vindicate his proof of the Kepler conjecture. In fact, three graduate students in Europe (not Hales's own) are already at work on separate parts of this project, two using Isabelle and one using Coq. Hales expects them to finish in about 7 years.

But Hales thinks that computer verifiers have implications far beyond the Kepler conjecture. "Suppose you could check a page a day," he says. "At that point it would make sense to devote the resources to put 100,000 pages of mathematics into one of these systems. Then the mathematical landscape is entirely changed." At present, computer assistants still take a lot of time to puzzle through some facts that even an advanced undergraduate would know or be able to figure out. With a large enough knowledge base, that particular time sink could be eliminated, and the programs might enable mathematicians to work more efficiently. "My own experience is that you spend a long time going over and going over a proof, making sure you haven't missed anything," says Carlos Simpson, an algebraic geometer and computer scientist at the University of Nice in France. "With the computer, once it's proved, it's proved. You only have to do it once, and the computer makes sure you get all the details."

In fact, computer proof assistants could change the whole concept of proof. Ever since Euclid, mathematical proofs have served a dual purpose: certifying *that* a statement is true, and explaining *why* it is

Have a Coq and a Smile

Why would hundreds of computer scientists devote more than 30 years to developing mathematical proof assistants that most mathematicians don't even want? The answer is that they are chasing an even more elusive grail: self-checking computer code.

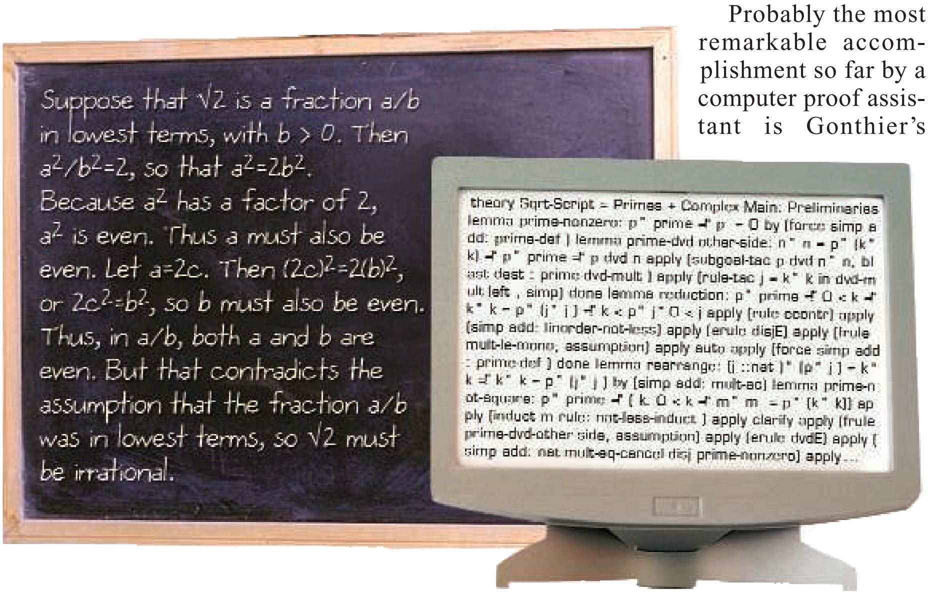
In a sense, the statement "this program (or chip, or operating system) performs task *x* correctly" is a mathematical theorem, and programmers would love to have that kind of certainty. "Currently, people who have experience with programming 'know' that serious programs without bugs are impossible," Freek Wiedijk and Henk Barendregt, computer scientists at the University of Nijmegen in the Netherlands, wrote in 2003. "However, we think that eventually the technology of computer mathematics ... will change this perception."

Already, leading chip manufacturers use computer proof assistants to make sure their circuit designs are correct. Advanced Micro Devices uses a proof checker called ACL2, and Intel uses HOL Light. "When the division algorithm turned out to be wrong on the Pentium chip, that was a real wake-up call to Intel," says John Harrison, who designed HOL Light and was subsequently hired as a senior software engineer by Intel. **–D.M.**

Published by AAAS

true. Now those two epistemological functions may be divorced. In the future, the computer assistant may take care of the certification and leave the mathematician to look for an explanation that humans can understand. "Just because a proof is explanatory doesn't mean it's certain," says Harvey Friedman, a logician at Ohio State University in Columbus. "Just because it is certain doesn't mean it's explanatory. They are two separate dimensions."

So far, Hales, Simpson, and Friedman are part of an extremely small minority: mathematicians who have taken the trouble to learn about proof assistants. "Mathematicians don't know about [computer proof verification], they're not interested in it, and they don't believe it," says Freek Wiedijk, a computer scientist at the University of Nijmegen in the Netherlands who specializes in proof verification. Simpson says much of the mistrust may stem from a misimpression that computerized proof checkers are trying to automate mathematical creativity.



Suppose that √2 is a fraction a/b in lowest terms, with b > 0. Then $a^2/b^2=2$, so that $a^2=2b^2$. Because a^2 has a factor of 2, a^2 is even. Thus a must also be even. Let a=2c. Then $(2c)^2=2(b)^2$, or $2c^2=b^2$, so b must also be even. Thus, in a/b, both a and b are even. But that contradicts the assumption that the fraction a/b was in lowest terms, so √2 must be irrational.

```
theory Sqrt-Script = Primes + Complex Main: Preliminaries
lemma prime-nonzero: p " prime ⟹ p ≠ 0 by (force simp a
dd: prime-def ) lemma prime-dvd other-side: n " n = p " (k "
k) ⟹ p " prime ⟹ p dvd n apply (subgoal-tac p dvd n " n, bl
ast dest : prime dvd-mult ) apply (rule-tac j = k " k in dvd-m
ult left , simp) done lemma reduction: p " prime ⟹ 0 < k ⟹
k " k = p " (j " j ) ⟹ k < p " j " 0 < j apply (rule ccontr) apply
(simp add: linorder-not-less) apply (erule disjE) apply (frule
mult-le-mono, assumption) apply auto apply (force simp add
: prime-def ) done lemma rearrange: (j ::nat ) " (p " j ) = k "
k ⟹ k " k = p " (j " j ) by (simp add: mult-ac) lemma prime-n
ot-square: p " prime ⟹ ( k. 0 < k ⟹ m " m = p " (k " k)) ap
ply (induct m rule: nat-less-induct ) apply clarify apply (frule
prime-dvd-other side, assumption) apply (erule dvdE) apply (
simp add: nat mult-eq-cancel disj prime-nonzero) apply...
```

Different languages. Machine proofs (right) can look very different from "human" versions.

In fact, an assistant can no more prove the Four Color Theorem than an online thesaurus can write *Hamlet*. In a typical session with a proof verifier such as Isabelle or Coq, the mathematician enters the hypotheses at the top of the computer screen and the "proof obligation"—the conclusion—at the bottom. She decides on a "tactic" to simplify the proof obligation—for example, subdividing it into simpler cases, performing a calculation, or applying a previously known theorem. Each time the user enters a tactic, the computer program executes it and updates the proof obligation. When there are no more obligations left, the proof is verified.

One stumbling block is that published proofs never specify every step. Every math student is familiar with the dreaded words, "it is obvious that …" To the computer, *nothing* is obvious. It is up to the user to break the "obvious" step down into subtasks that the computer can check. Diagrams are particularly troublesome; the user must somehow parse the pictorial information into allowable tactics.

All in all, people who have used proof verifiers say they can formalize about a page of textbook mathematics in a week. Avigad says he reached a top speed of a page a day while working on the Prime Number Theorem—close to the break-even point at which it will be worth mathematicians' time. "When it becomes not too much harder to formally verify a proof than to write it up carefully, it starts looking like a win," he says.

Probably the most remarkable accomplishment so far by a computer proof assistant is Gonthier's recently completed verification of the Four Color Theorem. This theorem began as a conjecture in 1852, when a graduate student at University College London named Francis Guthrie asked his professor Augustus DeMorgan if he could prove that any map can be colored with four colors in such a way that no two adjacent countries have the same color. After more than a century of unsuccessful attempts, some by eminent mathematicians, two computer scientists, Kenneth Appel and Wolfgang Haken, finally proved it in 1976. Their computation-intensive argument raised an immediate furor. "Mathematicians over 40 years old couldn't be convinced that a proof by computer was correct, and those under 40 couldn't be convinced that a proof with 700 pages of hand calculations was correct," jokes Robin Wilson, a graph theorist at the Open University in Milton Keynes, U.K. Enough questions remained about its validity that another team of graph theorists, led by Paul Seymour of Princeton University and Neil Robertson of Ohio State University, published a revised proof in 1995. Even this streamlined proof relies on a case-by-case analysis of more than a billion different maps, far more work than a human mathematician could do in a lifetime. The computer did it in 3 hours.

Crunching through special cases also played a large role in checking the proof, Gonthier says. "From the point of view of someone using a formal computation system, those are really the easy parts. The hard part, in this case, was finding formal definitions that captured correctly the intuitions behind graph theory." Gonthier had to revamp Seymour and Robertson's approach considerably, so that the proof assistant would understand what elementary ideas such as "the next edge on the left" meant. In his final proof script, he estimates that 19,000 lines came directly out of the Robertson and Seymour paper, and 19,000 lines were his own work. (Another 22,000 lines are white space, comments, and "infrastructure.")

Specialists in computer verification give Gonthier's work very high marks. "That guy is amazing," says Wiedijk. "I can't compete with this kind of genius." Hales calls it "a magnificent piece of work. What this means is that the proof is finally self-checking. You don't have to worry about whether the programmers introduced bugs into the computer code." On the other hand, every graph theorist contacted for this article either had not heard of Gonthier's work or remained skeptical about it. "I have no serious doubts that computers have done their part flawlessly," says Bojan Mohar, a graph theorist at the University of Ljubljana, Slovenia. "[But] I cannot confirm that Gonthier has made the correct translation of the [human] proof into computer form." Others doubt the machines themselves. Coq may tell them that Gonthier's code is correct, but why should they trust Coq?

"It's reasonable to say [Coq's code] has been verified experimentally," Gonthier says. Coq is a program that has been developed (at INRIA in Paris) over a period of 20 years, boasts a community of about 100 active users, has source code that is open for inspection, and runs on several different computers and operating systems. Besides, he argues, "even traditional mathematical proofs use physical artifacts. You're relying on the fact that when you flip back to a previous page, the ink doesn't change. Your day-to-day experience is that the ink doesn't change. Similarly, our experience with computers is that once given a consistent set of instructions, they compute consistently. It's just hard to give them a consistent set. Proof-assistant technology makes sure that you do."

–DANA MACKENZIE

Dana Mackenzie is a freelance writer in Santa Cruz, California.

Published by AAAS