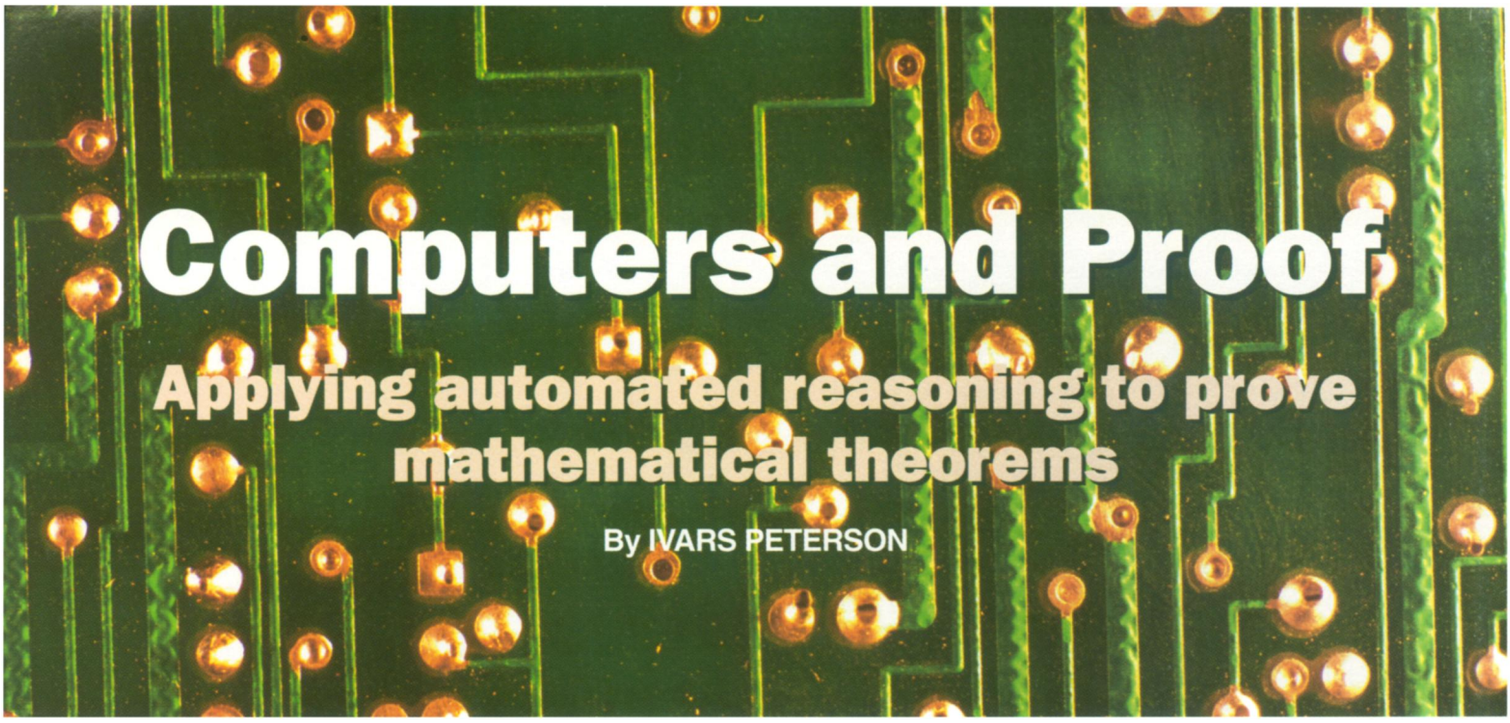

Computers and Proof

Applying automated reasoning to prove mathematical theorems

By IVARS PETERSON

The late Paul Erdős liked to describe a mathematician as "a device for turning coffee into theorems."

To Erdős, however, it wasn't enough to cobble together any old proof to establish the truth of a conjecture. He subscribed to the notion that God has a book containing all the theorems of mathematics, together with their most elegant and beautiful proofs. The goal of mathematicians was to uncover these sublime instances of logical reasoning.

When Erdős wanted to express particular appreciation of a proof, he would exclaim, "This is one from the book!"

One result unlikely to qualify for "the book" is last fall's proof of the so-called Robbins conjecture, which concerns an aspect of the basic rules of logic. Nonetheless, the proof is noteworthy because it was found by a computer program, succeeding spectacularly where mathematicians had failed.

Originally proposed in the 1930s by Herbert Robbins, now at Rutgers University in New Brunswick, N.J., the problem had stumped everyone who tackled it over the years. It finally succumbed to a computer program designed to reason in a general way rather than a program designed to solve a particular problem, whether an equation or the moves of a chess game.

Called EQP for "equational prover," the automated reasoning software was developed by computer scientist William McCune of the Argonne National Laboratory in Illinois.

"I was surprised that the problem finally yielded to the program because some powerful minds had tried to solve it and hadn't succeeded," says mathematician Larry Wos, a colleague of McCune's at Argonne. "It's an impressive result. We've made a lot of progress in automated theorem proving in the last few years."

"It's a real breakthrough," says mathematician Stanley N. Burris of the University of Waterloo in Ontario. "I had no expectations that a proof would emerge at this time."

McCune's computer-generated proof is to be published in a future issue of the JOURNAL OF AUTOMATED REASONING.

Computers are not new to the business of proving theorems. In the past, however, they generally served as bookkeepers or assistants to check the large number of special cases needed to establish a theorem (SN: 12/24&31/88, p. 406). Mathematicians or computer scientists outlined the necessary steps, wrote the special-purpose software, and specified the calculations required for a proof.

"What we do is quite different," McCune says. Though the computer's search has constraints, neither the path to a result nor the ultimate outcome is determined beforehand. Indeed, because typically no one knows any route to a solution, the search can't be guided in a meaningful way.

Wos and his coworkers began developing software for automated reasoning in the 1960s. They focused on how to describe a mathematical problem in terms that a computer can handle and how to get it to prove theorems by drawing conclusions that follow inevitably and logically from given postulates, or axioms.

"We never tried to imitate what people do," Wos insists.

The idea was to give an automated reasoning system the statement of a conjecture along with a few rules of thumb constraining the search among the infinite number of possible logical paths to a proof, then allow it to proceed on its own.

Advances came in the form of improved strategies for keeping the computer from getting entangled in lengthy chains of reasoning that apparently led nowhere. Those improvements, along with faster computers and better software engineering, eventually produced an agile reasoning system called Otter, aimed at questions in abstract algebra and formal logic.

Using Otter, McCune and his coworkers have in recent years solved a wide variety of mathematical problems, generating original proofs of theorems in logic, algebraic geometry, group theory, and other areas of mathematics.

A few years ago, McCune decided to make a fresh start and develop a "daughter of Otter," incorporating techniques and strategies that researchers had found useful while experimenting with Otter during the previous decade. "Otter was just getting too big and too patched-up," McCune explains. "I wanted a new program that was easier to modify to try things out."

The result was EQP. The program incorporated a number of strategies for selecting possible search paths, including simply following the path defined by the smallest equation or formula at certain steps or side-stepping expressions containing more than, say, 100 logical relationships.

Initial tests of EQP were encouraging, and the program succeeded in proving, sometimes for the first time, several theorems of special interest to a few logicians and other experts. These results, however, were of limited value to

most mathematicians.

"The trouble was that [McCune and his coworkers] had never really done a mainstream problem of sufficiently wide interest," Burris remarks. Until that happened, "mathematicians would have no appreciation of how powerful automated theorem proving could be."

As a benchmark test of his system, McCune turned to the Robbins conjecture, which states that a set of three equations in logic is equivalent to a Boolean algebra (see box). In other words, the problem is to determine whether this particular set of equations incorporates the basic laws of Boolean logic, which specify the different ways in which collections of objects can be combined or manipulated using logic operations such as "and," "or," and "not." Such logic underlies the workings of today's digital computers and is often used to facilitate database searches.

Robbins himself worked on the problem, and it was later taken up by others, including the late Alfred Tarski, a prominent logician at the University of California, Berkeley. When Tarski failed to make any headway, he handed the problem out as a challenge to graduate students and visitors.

Wos first came across the problem in 1979, when he helped a student develop a novel way of attacking it by working backwards—that is, by finding conditions that, if true, would prove the theorem. That promising start, however, did not lead to an immediate proof.

At the same time, the problem had characteristics ideal for an automated attack. Embodied in just three equations, the Robbins conjecture could be stated succinctly in a form a computer could understand. Moreover, researchers could identify a number of conditions that, if true, would each point to a complete proof.

"We tried it many times using Otter and other programs," Wos says. No luck.

Last fall, McCune's EQP was ready for a new challenge. McCune let the prover software grapple continuously with the Robbins conjecture. He checked the computer every day for progress.

On the eighth day, the computer found the answer and stopped. It had proved that the Robbins conjecture is true.

Fresh out of the computer, the proof consisted of a cryptic chain of lengthy, practically unreadable statements of logic. McCune then refined EQP's procedures to obtain somewhat more streamlined proofs.

In the meantime, several mathematicians independently checked the computer proof and found it correct. Burris, for one, converted the proof's long statements into simpler forms that could be more readily understood by humans. "I ended up with a lot of little equations," he says. "You could easily sit on a bus and go through the hundred or so steps of the proof."

Philosophy graduate student Branden E. Fitelson of the University of Wisconsin-Madison also worked through the result. "It's a very complicated proof, not at all elegant or conceptual," he comments. "It's a perfect example of the type of proof that a machine can find but we can't because of its complexity."

EQP's success raises some interesting questions about mathematical creativity. If a mathematician had proved the conjecture, the achievement would have been heralded as a significant accomplishment. Yet it's highly unlikely that a mathematician would have come up with the kind of proof that EQP found.

At the same time, mathematicians generally have no way of telling beforehand—just by looking at a problem or a conjecture—what kind of attack will subdue it. In fact, one of the attractions of the Robbins conjecture was that it looked vulnerable to techniques mathematicians had used to prove similar theorems, differing only in what seemed like minor details.

"There's no general theory of when [various methods] are not going to work," Fitelson says. "That makes life interesting, but it's also hard to predict how things are going to work out."

No one knows whether the computer proof generated by EQP represents a fluke success or the first of many triumphs.

"That's something we'll be able to judge only in hindsight," Burris says. "Though we have strong theorem provers, we don't yet know what power is required to be really competitive with human beings."

Nonetheless, he adds, "we now know [automated theorem proving] can do something significant, and we didn't know that a year ago."

McCune, Wos, and others are taking a closer look at EQP's performance to see if they can learn something that could help with comparable problems. "What was it doing during those 8 days?" Wos wonders. "Is there some kind of strategy that would have tightened up the reasoning?"

The researchers are also interested in extending the kind of deductive reasoning an automated theorem prover can apply from relatively simple logical relationships involving equality to more complicated expressions. They would like to introduce mathematical induction, which would allow the computer to make inferences.

"Our programs do not learn, do not make judgments, and they do not invent concepts," Wos notes. Nonetheless, it may be possible to program them to be self-analytical—to recognize in a mechanical sense when searches are not going well and to shift strategies.

McCune and Wos are also trying to interest mathematicians in suggesting problems that could be tried out on EQP or Otter. McCune, for example, is already working with R. Padmanabhan of the University of Manitoba in Winnipeg, using reasoning software to establish a large number of simple, unproven conjectures as part of a larger effort.

"Those conjectures are the kind of thing that might take a mathematician an hour to do by hand but only a few seconds with a theorem prover," McCune says.

Wos and McCune envision offering mathematicians the option of using a computer to free them from the drudgery of finding proofs, allowing them to spend more time on harder, more interesting problems.

"Automated reasoning could complement your reasoning, so you could freelance more and use your intuition more," Wos argues.

Programs such as Otter and EQP already work well in specialized areas of pure mathematics, such as the theory of algebraic systems. "I think that computer-generated proofs will have a significant influence in the near future [in those fields]," says mathematician Kenneth Kunen of the University of Wisconsin-Madison.

Similar approaches could help engineers design circuits or allow computer programmers to verify the correctness of their software. Indeed, research groups throughout the world are working on such applications of automated reasoning.

Computers have yet to come up with a proof that belongs in "the book," and they can't turn coffee into theorems. However, the power of theorem-proving programs continues to increase.

"Can machines reason at the same level as humans?" Burris asks. "In the long run, I would bet on it, but it's unlikely I'll see it in my lifetime."

Boolean logic

Boolean algebra is a mathematical model of some of the basic rules of logic. It includes such laws as "for any proposition P, the negation of the negation of P means the same thing as P" or "for any two propositions P, Q, the conjunction of P and Q is false if and only if one or both of them is false."

The Robbins problem is equivalent to proving that the equation not(not(P)) = P can be derived from the following three equations:

P or Q = Q or P;

(P or Q) or R = P or (Q or R);

not(not(P or Q) or not(P or not(Q))) = P.