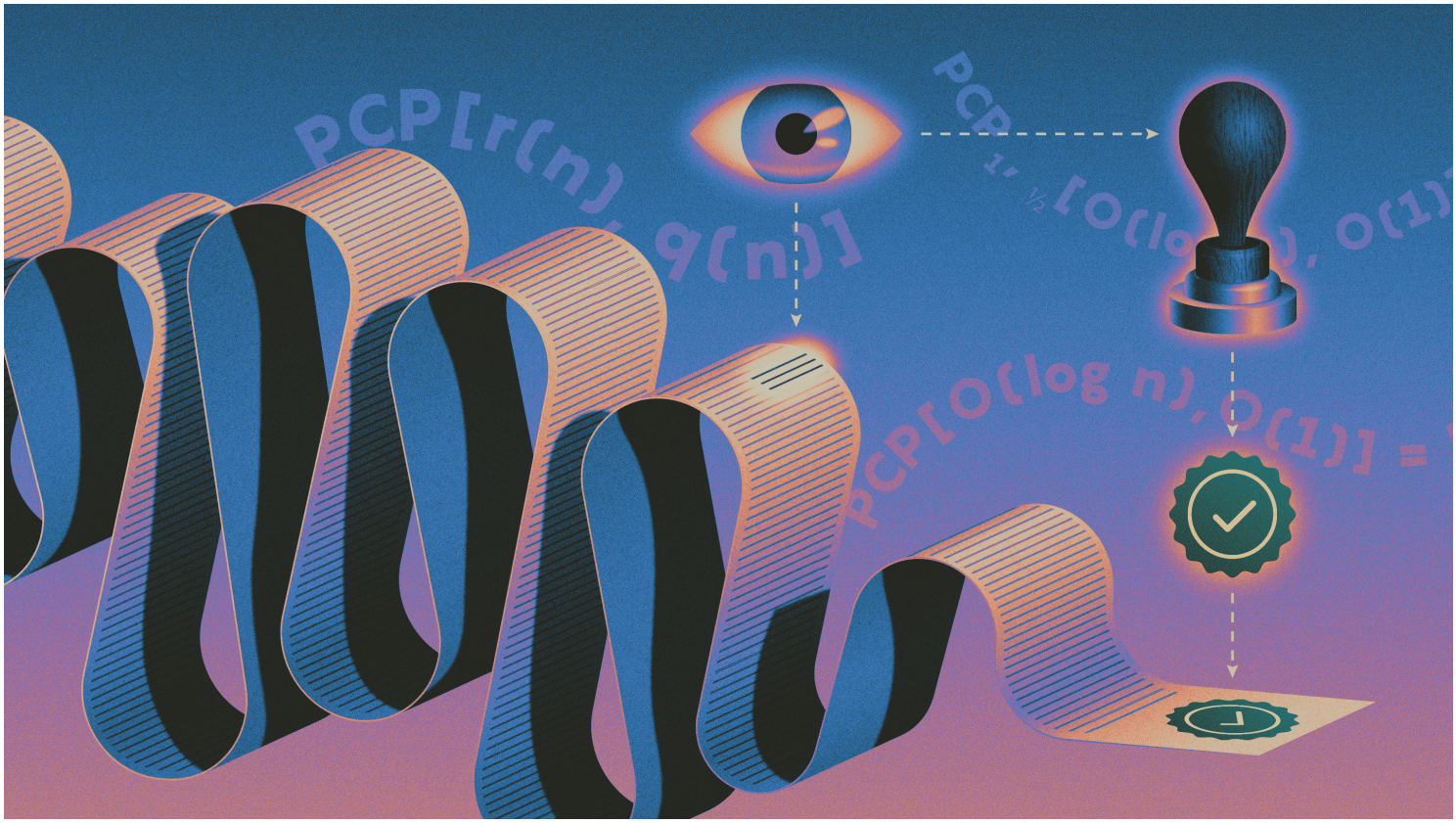
COMPUTATIONAL COMPLEXITY

How Computer Scientists Learned to Reinvent the Proof

By MORDECHAI RORVIG

May 23, 2022

Why verify every line of a proof, when just a few checks will do?



Kristina Armitage/Quanta Magazine

If a million computer scientists had dinner together, they'd rack up an enormous bill. And if one of them was feeling particularly thrifty and wanted to check if the bill was correct, the process would be straightforward, if tedious: They'd have to go through the bill and add everything up, one line at a time, to make sure that the sum was equal to the stated total.

But in 1992, six computer scientists proved in two papers that a radical shortcut was possible. There's always a way to reformat a bill — of any length — so that it can be checked with just a few queries. More importantly, they found that this is true for any computation, or even any mathematical proof, since both come with their own receipt, so to speak: a record of steps that a computer or a mathematician must take.

This remarkably succinct format was called a probabilistically checkable proof (PCP). (According to Ryan O'Donnell, the acronym was apparently chosen after police mistakenly went to the hotel room of one of the inventors, Shmuel Safra, while trying to make a possible drug bust for phenylcyclohexyl piperidine — also known as PCP, or angel dust.)

PCPs have become some of the most important tools in theoretical computer science. Recently, they've even found their way into practical applications, such as in cryptocurrencies, where they are used for rolling up large batches of transactions into a smaller form that is easier to verify.

"I don't know of any example — or maybe it's just very rare — of such deep algebraic tools actually making it into practice," said Dan Boneh of Stanford University.

Prior to the creation of PCPs, computer scientists had identified a whole class of problems with solutions similar to dinner checks — easy to verify, once you have it. But for many of these problems, finding a solution in the first place seemed to take an impractical amount of time.

Computer scientists named this class of hard-to-solve but efficient-to-verify problems NP. It provides a conceptual home for many of the practical problems we care about and also for much more abstract problems, like finding proofs of mathematical theorems. Proofs are step-by-step recipes that establish their mathematical conclusion with absolute certainty — just as an itemized bill provides proof of the total owed. Proofs can be hard to obtain, but once you have one, it will be straightforwardly checkable. That puts proofs squarely in the category of NP.

In the 1980s, computers scientists began reimagining what proofs could be. Cryptographers wondered whether it might be possible to know that a proof was true without seeing its contents. They split up the structure of a proof into a two-part system, with a "verifier," who tries to check a proof, which is provided by a "prover," who somehow finds the proof.

The PCP theorem of 1992 showed that it was always possible for the prover to encode a proof in a new form, such that it could be verified with a constant number of queries, regardless of the proof's original length. That number of necessary queries was eventually brought down to just two or three. The verifier would not know it was true with perfect certainty, but by making more queries, it can steadily and straightforwardly increase its certainty. The result defied intuition. Surely much longer proofs would require you to examine more evidence? Not so.

"It was inconceivable that such a thing was true," said Swastik Kopparty of the University of Toronto. "Until shortly before the PCP theorem was proved, nobody could even dare to make such a statement."

The theorem gave a new understanding of NP and explained some of its intriguing properties. Computer scientists had found that for some NP problems, answers seemed not only hard to compute, but hard to approximate as well. The PCP theorem helped explain why. It said that if a solution to an NP problem was found, it could always be reformatted in a way where most checks from a verifier (say 90 percent) would pass (but not all of them, because the proof is still just probabilistic). From the vantage point of the verifier, it would therefore look like the problem was solved approximately, to 90 percent accuracy. But because NP problems are hard to solve, it is often difficult to find a PCP for them, and therefore it's equally hard to find a solution that is approximately correct beyond a certain point (such as 90 percent).

Scientists also began thinking about the possibility of practical applications for PCPs. Unfortunately, the PCPs of the '90s were grossly inefficient. It would take thousands of years for a prover to produce a PCP representing any practical computation. Further, any actual PCP would be huge, requiring a hard drive the size of a planet.

But by 2008, researchers had found PCPs whose size and computation scaled up much more slowly, making them more manageable. Then, in the mid 2010s, researchers began to build new forms of PCPs that were even smaller, called interactive oracle proofs (IOPs), which added additional rounds of interaction between the prover and verifier, where in each round, the prover could produce something much smaller more quickly.

"By adding interaction and using a lot of the same math that was ported over from PCP technology, you get practical stuff," said Eli Ben-Sasson, a computer scientist who left the Technion in Haifa, Israel, to start the company StarkWare.

In the past decade, computer scientists also found direct applications for PCPs (and their descendants) in the software behind cryptocurrencies, which are now raising intriguing theoretical questions of their own. In one of these software systems, a large computer (the prover) validates financial transactions and places the validation computation into a PCP, so that a smaller computer (the verifier) can validate the transactions much faster.

But suppose the prover is trying to cheat, for example, by concealing a set of false transactions within the PCP. When a PCP system (consisting of prover, verifier, and PCP) is resilient against this kind of deception, researchers say it has "soundness." Soundness is both important in theory as well as in practical applications, where better soundness (quantified by a certain parameter) translates to faster verification and less computational work.

A paper released in May 2020 by Ben-Sasson, Dan Carmon, Yuval Ishai, Kopparty and Shubhangi Saraf showed that the soundness for one modern PCP system reaches a fundamental limit from theoretical computer science. This is the maximum known durability of data when it has been encoded in one classic form, known as a Reed-Solomon encoding, which is also the way that a proof or computation is encoded by a PCP.

PCPs can still be made more efficient. Recently, two groups of researchers discovered an optimal method for encoding a large block of data such that checking it at just a few spots reveals if the whole block has been corrupted. This method does something similar to a PCP by providing a test of integrity that depends on just a few queries, while also reaching perfect efficiency in terms of speed and size. Researchers see it as a proof of concept that one day perfectly optimal PCPs might be found.

"It's not an easy problem," said Dana Moshkovitz of the University of Texas, Austin. "[But] it feels like we should just go and do it."