An Interview with Robert Soare

University of Chicago's Robert Soare, the Paul Snowden Russell Distinguished Service Professor of Mathematics and Computer Science, offers his reflections on Alan Turing.

Interviewed by Arefin Huq

DOI: 10.1145/2090276.2090284

AREFIN HUQ: Your upcoming book is entitled, *Computability Theory and Applications*, and it's subtitled, "The art of classical computability." Why did you choose that title?

ROBERT SOARE: Mathematicians value work on the basis of beauty. It has to be correct, but the question is: Is it beautiful? Is it a beautiful theorem, a beautiful proof? At the University of Chicago no one asks me to do the mathematics of building bridges on a deadline. I choose the problems that I work on, and I choose problems for my students, and they work on the things they enjoy doing. We do it because we hope to solve important problems but we do it also because it's beautiful. So the subtitle, "The art of classical computability," is supposed to represent this thought.

AH: What makes Turing's model of computation so compelling?

RS: Why do we give so much credit to Turing? His advisor, Alonzo Church, was already a full professor before Turing even started out, and Church proposed one definition of computable function in 1934, the lambda-definable functions, and another one in 1935. Church got it right, twice, and he got it first. He got it right in the sense that he proposed a model which later proved to be correct. Turing came on the scene only in 1936, when he published his paper after Church's paper was already announced and published. And yet if you pick up any book on computability now you'll see a description of a Turing machine. You won't see the lambda-definable functions of Church and you won't see the recursive functions of Herbrand-Gödel, but you will see Turing's model because it is easy to understand. That's the first part.

Robert Soare

The second part is that there are famous problems in mathematics, like the Riemann Hypothesis, like the Poincaré Conjecture that was done recently, like Hilbert's Tenth Problem. All these are specific, concrete, well-defined mathematical problems and if someone solves one of these he becomes famous. Solving means to give a mathematical proof that other mathematicians can verify as a correct proof.

Hilbert had proposed in the 1920s that people find an algorithm for deciding statements in the logical language of mathematics. If accomplished it would have meant that, at least in theory, much of the work of mathematicians could have been eliminated, putting many mathematicians mostly out of business. So Church and Turing suspected this was not possible. But to get an undecidable problem they had to find a way of defining and listing all computable functions and then diagonalizing by finding a problem that is not solved by any one of those computable functions.

But the first task was to define a computable (i.e. calculable) function and then prove that it captured what a human being could compute or calculate. Gödel, who was the most famous person in mathematical logic after 1931, questioned whether it was possible to do this. He suspected it was an abstract, informal notion that couldn't be captured. Gödel stated that whether all effectively calculable functions are recursive, "cannot be proved, since the notion of finite computation is not defined, but serves as a heuristic principle."

For 3000 years before this, human beings had been calculating—Euclid, Archimedes, Leibniz, and so on. A calculation meant you write down things on a piece of paper or a slate or whatever you have in front of you and you move to the next step and the next step and so forth. That's what a calculation represented for human beings. What Turing said was, well, the human being could calculate with an infinite number of symbols, but if he did, those symbols could be coded in binary, 0 and 1. So therefore we only really need two symbols. Next he said that computation can be carried out in three-dimensional space: Archimedes could have had shelves and he could have put part of the computation on the top shelf and part on the middle shelf and so forth. But we can code that by a one-dimensional tape,

Photo courtesy of Flickr user blond avenger, Lo van den Berg

because we can code all the places and the space and put them on the tape. Also we can assume that there are distinct squares and each square holds a single symbol. One by one he went through all the things that a human could do and he showed how his machine could do it. So number one, he came up with a simple, understandable model, and number two and perhaps even more important, he then gave a proof, not a formal mathematical proof, but a demonstration that anything that could be calculated by a human being could be done by one of his machines.

AH: You've compared Turing to Michelangelo. Please explain.

RS: Around 1450, Donatello, a famous sculptor, sculpted a bronze statue of David, the David who killed Goliath. It was a typical pose at the time. David had on a helmet and he was holding a sword, and at his feet was the head of Goliath whom he had slain. This was a remarkable piece of sculpture. It was the middle of the Renaissance. It influenced Michelangelo and other people, and then it was succeeded in 1475, 25 years later, by Verrocchio.

Verrocchio was well known as the teacher of Leonardo da Vinci and Botticelli, and he sculpted another bronze statue in a similar pose. Helmet on the head, Goliath's head on the ground under David's foot, and David with his sword. The pose was a little different and the expression on his face was very complex. This again is a remarkable sculpture, one that's highly prized around the world, and was an advance over the 1450 Donatello. Then in 1501 to 1505 Michelangelo got a piece of pure Carrera marble that was taken from a nearby quarry. It had already been started by someone else and had some defects, so he had to work around the defects, and out came Michelangelo's David. Michelangelo's David stands in a gallery in Florence called the Accademia. It's gleaming white as you come in and it's remarkable because the human figure is much less stiff, much more vibrant, than in the preceding two. Then he has the famous *contrapposto*. That means the upper part of the body tilts to one side, the lower part of the body tilts to the other side, and so it gives a feeling of motion and a feeling of the human body displaying its muscles.

So my feeling is that David was just a kid. He had not even yet reached puberty. He was a shepherd boy and the Israelites were facing the Philistines. The Philistines put forward their best warrior, Goliath, a huge giant, and they demanded one-on-one conflict of the best warrior against the best warrior. The Israelites were all afraid to face this huge giant, and along comes this little kid, a kid with a slingshot, and he steps out and he slings the stone, and the stone hits Goliath in the head and kills him.

So Michelangelo displays David not standing on the head of Goliath—you don't even see Goliath's head—and displays David, not with a sword, but with a sling over his shoulder. Two things about the statue are remarkable. Number one, it's an enormous advance over everything that had gone before, and number two, it says something new about human beings and the human condition. David is standing there with tension on his face before he meets Goliath, not afterwards as the others who are triumphant and gloating over the death of Goliath. He's about to go out with the sling on his shoulder. David and Turing had a mental triumph, not a physical one.

Turing was only 23 years old when

he heard of the problem. He was just a kid. He went into the lectures in 1935 of Max Newman, who was a topologist at Cambridge. He heard the lectures, got interested in this problem, went away, lay down in a meadow, and then in the meadow he got the idea for what to do. He came back and gave his solution to the astonished Max Newman in 1936. So Turing was a kid with a slingshot. He was facing a phalanx in Princeton of Gödel, the most famous person in mathematical logic in the world, whose Incompleteness Theorem revolutionized our idea of proofs in mathematics in 1931. The group also included Alonzo Church, who had been working on this problem for a decade, his students Kleene and Rosser, and Emil Post, who was nearby in New York and had been working on this stuff. All these people came up with various ways of trying to approach computability. Turing, with a slingshot and a single stone, hit the mark.

AH: You've recommended Turing's 1936 paper and Post's 1944 paper as recommended reading for all computer science students [1, 2]. Why?

RS: At the University of Chicago in the 1930s they had something called the "Great Books" program, where people were supposed to read the original books—the ancient books of Greece and Rome, Shakespeare, and the most fundamental and important books of Western literature. So I thought a corresponding thing that would make sense is the "Great Papers" program. Each paper should be readable: Its exposition should be beautiful in the sense we just described. So Turing 1936 and Post 1944 satisfy that. I have two criteria in recommending these papers. First of all it should be fundamental to the subject, then and now, and second it should be very well written so that it's intuitive, easy to understand and appealing.

AH: How is the field of mathematical logic relevant for the future?

RS: I think it's very relevant to the future. Logic was not so central in 1935 when Church and Turing and the others came around, but then it's interesting that the constructs in logic like lambda computability, recursive functions, Post canonical systems, and so forth, have all had important implementations in actual computer science. The lambda calculus was used by another Princeton graduate in mathematics, John McCarthy, who created the computer language Lisp using the lambda calculus. McCarthy is considered by many to be the father of artificial intelligence. He completed his Ph.D. at Princeton in 1951 under Solomon Leschetz and spent much of his career at MIT and Stanford. He created Lisp in order to create Turing machines in the limited computing environment at his disposal. The lambda calculus has also been used in programming languages and other parts of modern computer science. The 1943 paper by Post had something called normal systems and Post canonical systems, which were then taken up by the generative grammars of Noam Chomsky at MIT. So logic has had a lot of influence.

There's a researcher at Microsoft, Yuri Gurevich, who gave a talk in which he said engineers study calculus, which they don't use, and they do not study logic, which they do use. My career began in about 1960 when I took my course from Church and the whole thing has just mushroomed since then in terms of the connections with other fields.

I think a good analogy right now is Turing. If you look at the second category of papers on my Web page, the bottom one (not my own paper) is an article in the *Princeton Alumni Weekly* listing Turing as the second most influential alumnus of all time after James Madison. That puts him ahead of Woodrow Wilson and John Foster Dulles and a whole lot of other people. That's an amazing statement. I didn't think that people even knew who Turing was, but this was judged by some independent external committee. The celebration that's going on for Turing now is just enormous. It's a major event. There's just a mushrooming interest in Turing and his work and his legacy that is indicative of an appreciation for the role of logic and computability in our society.

AH: What advice do you have for students getting started in research today?

RS: Entering graduate school is like entering a guild. You enter a guild—a medieval guild like shoemaking or blacksmithing—and you apprentice yourself to the main person in that field for a while. You learn the craft from him and you go deeper and deeper into that field, into that guild in your second, third, fourth, fifth year. And then that person helps you get a job in another village, part of the same overall guild but in another village, and it's mainly your advisor who finds you that job. The subfields in computer science are sufficiently developed and technical that it's difficult to get into them by yourself. You need an advisor, and he has to help you learn the material and help you choose problems to work on. Turing did it in 1936, yes, but nowadays it would be hard to work on problems that you just made up or that you saw in a paper. If you see them in a paper that means other people worked on them and couldn't get them. So the advisor is key.

I think there is a bright future for students both in the theoretical part and the more practical part of computer science, and I think those parts interact. If you're a computer science undergraduate or graduate student, don't think of yourself as necessarily only in the theoretical areas or only in the practical areas because there is an interaction between the two. At least in the beginning I would say: Take the courses, do as well as you can, and keep your mind open. Later on, by the third or fourth year, you tend to specialize in one particular area and that's fine, there's nothing wrong with that. Then my advice is: Choose your advisor carefully because the advisor is the one who plays the biggest role as far as your future and your dissertation. So choose an advisor who is going to take a strong interest in your work and help you get a job and things like that. But first, choose the topic that you like. So follow what you love doing, keep your mind open, and choose a good advisor.

Professor Soare has been invited to the Isaac Newton Institute at the University of Cambridge for four months to celebrate the Turing Centennial where he will lecture around the U.K. on Turing and computability.

References

[1] Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings London Mathematical Society*, Series 2 Volume 42, Parts 3 and 4 (1936), 230–265.

[2] Post, E.L. Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society* 50 (1944), 284–316.