DOI: 10.1145/2398356.2398357

Moshe Y. Vardi

Who Begat Computing?

The Turing Centenary with its furious pace is now behind us and we can afford some reflection on what has transpired. What started as an idea that the centenary of one of the founding figures of computing should be celebrated has turned into a global social phenomenon. A quick perusal of the Turing Centenary Web page (http://www.turingcentenary.eu/) reveals an amazing explosion of meetings, lectures, exhibitions, and volumes.

There is a risk, however, that in our focus on highlighting Turing's seminal contributions we may have gone from celebration to hagiography. Listening to so many speakers extol Turing's accomplishments, one could end up believing that Turing single-handedly begat computing, being the father of computability, universal machines, stored-program computers, cryptoanalysis, and artificial intelligence. This picture is simplistic and does not do justice to the richness of the story of how computing emerged between 1930 and 1950. We do not have one founding figure, we have several, and we should recognize and celebrate all of them.

The study of computability was launched at Princeton University, where Alonzo Church, together with his students Stephen Kleene and Barkley Rosser formalized computability in the early 1930s first in terms of the lambda-calculus, and then in terms of recursive functions (proposed by Jacques Herbrand and Kurt Gödel). They also proved the equivalence of the two formalisms, which led to Church's identification of computability with recursiveness. Yet, this characterization of computability was not compelling enough and described as "thoroughly unsatisfactory" by Gödel. It was then Turing's influential analysis of computability in terms of finite machines and its equivalence to the lambda-calculus and recursiveness that led to our current accepted understanding of computability, referred to as the Church-Turing Thesis. (Emil Post independently formulated another notion of machines, which turned out to be equivalent to Turing machines.)

Turing was a leading scientist in deciphering the German Enigma code at Bletchley Park in the early 1940s. Yet, unlike his computability work, which was done independently of the Princeton effort, breaking the Enigma was a collective effort. To start with, Turing was building on previous work by Polish and British code-breakers. I.J. Good played a key role in the Bayesian statistical analysis of Enigma messages and Gordon Welchman made key contributions to the design of the Bombe, the machine that used brute-force search to identify correct Enigma rotor positions. Overall, one must remember that the British code-breaking project was a huge effort; 12,000 people toiled at Bletchley Park during the war.

The claims that Turing invented the stored-program computer, which typically refers to the uniform handling of programs and data, are simply ahistorical. One can trace the kernel of the idea of handling programs and data uniformly back to Gödel's arithmetization of provability in 1931. The idea then showed up again in the lambda-calculus, recursive functions, and Turing machines. Turing invented a universal machine, a machine that can simulate all other machines, but he was preceded by the Princeton group, who constructed a universal lambda-term and a universal recursive function. While these ideas undoubtedly influenced the efforts of John von Neumann and his collaborators at the University of Pennsylvania in the 1940s, we should not confuse a mathematical idea with an engineering design. It was the EDVAC Report of 1945 that offered the first explicit exposition of the stored-program computer. Turing's ACE Report, which elaborated on this idea and cited the EDVAC Report, was submitted in early 1946. The first embodiments of the stored-program computer were the Manchester Baby and the Cambridge EDSAC, put into operation in 1949 and preceding the Pilot ACE, which was based on Turing's design and first run in 1950.

Turing was not the first to think about artificial intelligence (AI). The philosopher Charles S. Peirce wrote in 1887: "Precisely how much the business of thinking a machine could possibly be made to perform, and what part of it must be left to the living mind is a question not without conceivable practical importance." Nevertheless, Turing's 1950 paper "Computing Machinery and Intelligence" is indeed the first deep philosophical investigation of the possibility of artificial intelligence. While the Turing Test, referred in the paper as the "Imitation Game," has been rather under-influential in the history of AI, Turing does deserve the credit for putting the question of general machine intelligence so squarely on the table.

Computing emerged during the 1930–1950 period because the time was right. Many people played key roles in this development; assigning precise credit is quite impossible. Turing was a great computing pioneer, and his place in the computing pantheon is secure, but he is not alone there.

Moshe Y. Vardi, EDITOR-IN-CHIEF