Alan Turing and the Other Theory of Computation

Lenore Blum
Carnagie Mellon University, USA

The two major traditions of the Theory of Computation have for the most part run a parallel non–intersecting course. On one hand, we have the tradition arising from logic and computer science addressing problems with more recent origins, using tools of combinatorics and discrete mathematics. On the other hand, we have numerical analysis and scientific computation emanating from the classical tradition of equation solving and the continuous mathematics of calculus. Both traditions are motivated by a desire to understand the essence of computation, of algorithm; both aspire to discover useful, even profound, consequences.

While those in the logic and computer science communities are keenly aware of Alan Turing's seminal role in the former (discrete) tradition of the theory of computation, most *still* remain unaware of Alan Turing's role in the latter (continuous) tradition, this notwithstanding the many references to Turing in the modern numerical analysis/computational mathematics literature. These references are *not to* recursive/computable analysis (suggested in Turing's seminal 1936 paper), usually cited by logicians and computer scientists, *but rather to* the fundamental role that the notion of "condition" (introduced in Turing's seminal 1948 paper) plays in real computation and complexity.

This talk will recognize Alan Turing's work in the foundations of numerical computation (in particular, his 1948 paper "Rounding–Off Errors in Matrix Processes"), its influence in complexity theory today, and how it provides a unifying concept for the two major traditions of the Theory of Computation.