Robin Milner 1934–2010

Verification, Languages, and Concurrency

Andrew D. Gordon
Microsoft Research

Robert Harper
Carnegie Mellon University

John Harrison
Intel

Alan Jeffrey
Bell Labs

Peter Sewell
University of Cambridge

A special session of the symposium pays tribute to Robin Milner; the session features talks by Robert Harper (Carnegie Mellon University), John Harrison (Intel), and Alan Jeffrey (Bell Labs), and is organised by Andrew D. Gordon (Microsoft Research) and Peter Sewell (University of Cambridge).

Robin Milner was born near Plymouth in England. He was educated at Eton College, Windsor, and went up to King's College, Cambridge, in 1954. In between, his national service was with the Royal Engineers, in Egypt.

After university he was first a school teacher and then worked in London as a programmer at Ferranti, an early British computer company. In 1963 he took a job as lecturer at City University, and started research on the side—he never studied for a PhD. In 1968, he moved to Swansea to become a research assistant. He was excited by Scott's lectures in 1969 in Oxford on domain theory. During this time he learnt about Floyd's work on verification, and got to know Tony Hoare, and started trying to verify programs.

After meeting Zohar Manna during a visit to Carnegie Mellon University, he got a job with John McCarthy at Stanford in 1971. There he built the theorem proving system Stanford LCF, which he would later substantially refine with the addition of a metalanguage (ML) for constructing proofs.

> The interesting thing there was, they were just looking for somebody who would do something practical. Scott had recently produced a kind of logic of domains, a logic of the hierarchy of domains, a type-theoretic hierarchy. This to me seemed to be something which was practical. That is what LCF came out of. Since I was so interested in human-assisted – or I would say machine-assisted – proof, this seemed to be a wonderful opportunity for a program-oriented logic. It really took off for me when I realised that I could write down the syntax of a programming language in this logic and I could write the semantics in the logic.
>
> (*An interview with Robin Milner* by Martin Berger, 3 September 2003)

Moving to Scotland, Robin worked at the University of Edinburgh 1973–1994, as a professor from 1984. During this period he co-founded the LFCS, the Laboratory for Foundations of Computer Science, and served as its first director. This was also a time of amazing scientific productivity, for which he became a Fellow of the Royal Society and received the ACM A. M. Turing Award:

> For three distinct and complete achievements: 1) LCF, the mechanization of Scott's Logic of Computable Functions, probably the first theoretically based yet practical tool for machine assisted proof construction; 2) ML, the first language to include polymorphic type inference together with a type-safe exception-handling mechanism; 3) CCS, a general theory of concurrency. In addition, he formulated and strongly advanced full abstraction, the study of the relationship between operational and denotational semantics.
>
> (Citation for *1991 ACM A. M. Turing Award*)

We can now add to this list of his contributions Robin's refinement of CCS into the π-calculus, a more general theory of concurrency that takes into account dynamic generation and communication of names, and also bigraphs, a theory of ubiquitous computing.

He returned to England to occupy the first established Chair in Computer Science at the University of Cambridge. He was a member of the Computer Laboratory from 1995 onwards, serving as Head of the Laboratory 1996–1999. During the last year of his life, he resumed his Edinburgh connection as a part-time professor.

All three strands of his work have been enormously influential. Machine-assisted verification is now feasible for realistic systems software and for deep mathematics; Robin's concept of tactics is widely used in interactive provers such as HOL, Isabelle, and Coq, as is his idea of relying on a type-safe programming language to restrict the part of a prover that must be trusted to a small kernel.

ML itself is also still in use and has had a big impact on other programming languages: Robin gave us practical parametric polymorphism, now incorporated into most modern typed programming languages, and a push towards strongly typed languages and the use of operational semantics for language definition and metatheory.

His theories of concurrency underpin much work on verification (complementing the theory of sequential computable functions of Church and Turing) and are being applied to modelling processes from business to systems biology; Robin gave us the word bisimulation (while always taking care to credit David Park with the idea), and, more generally, a clear understanding of the role of observational congruences and coinductive proof methods.

He has left the subject a collection of elegant and finely honed intellectual tools, reified in mathematics and in software.

Robin and Lucy married in 1963 and died within weeks of each other in March 2010. They are survived by two children and two grandchildren.

Robin was always an inspirational teacher and colleague, and an unassuming and warm-hearted man; he will be greatly missed.

POPL'11, January 26–28, 2011, Austin, Texas, USA.
ACM 978-1-4503-0490-0/11/01.