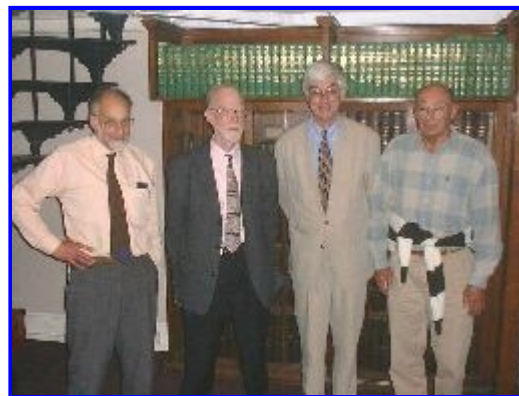


---

## Program Verification and Semantics: The early work

***Teresa Numerico and Jonathan Bowen***

---



On Tuesday 5 June 2001, a seminar on *Program Verification and Semantics: The early work* was held in the Director's Suite at the Science Museum, London. The seminar was organized with the co-operation of the British Computer Society (BCS) and the Computer Conservation Society (CCS). It was an instructive and enjoyable afternoon for the hundred or so people that attended the meeting.

Participating in the meeting were some of the pioneers and most important scientists in the fields of program verification and semantics and some of the most important historians of computing in Great Britain. It was a unique occasion that allowed the mingling of these two groups of people with an interest in computer science.

The organization of Prof. Jonathan Bowen, Prof. Cliff Jones and George Davis created a very good rapport between the audience and the speakers that presented their experiences in the field of formal methods. Presentations ranged from formal lectures to personal reminiscences. It was a historical event in itself: the special atmosphere allowed the audience to participate with interesting questions and reminiscences of their own.

After an introductory speech by Chris Burton on the aims of the CCS, Jonathan Bowen outlined very briefly the history of formal methods from Aristotle's logic to the use of Tony Hoare's assertions method in present debugging techniques, via Alan Turing's and Christopher Strachey's achievements.

The main speakers at the meeting were Sir Tony Hoare (Queen's University Belfast, Oxford University and Microsoft Research Cambridge), Joe Stoy (Oxford University Computing Laboratory), Prof. Robin Milner (Edinburgh and Cambridge Universities) and Prof. Peter Landin (Queen Mary, University of London).

Tony Hoare gave a talk on "*Assertions: a personal perspective.*" It was an excursus on his long and successful career, in and out of industry and academia. During his humanities, language and statistics training, he became interested in mathematical logic and its power, under the supervision of John Lucas. When he started his career in industry, at Elliott Brothers in 1960, he led a team with the aim of design and delivery of the first compiler for ALGOL 1960. According to him, his success was due to the fact the compiler used ALGOL itself as the design language. He became interested in axiomatic theory, reading Bertrand Russell's *Introduction to Mathematical Philosophy*, and realized that computer programs could also be expressed and defined using assertions, known as preconditions and postconditions relative to the results that were expected at the launch and termination of a program.

After his industrial experience, in 1968 he pursued his research into assertions in an academic setting at Queen's University in Belfast and from 1977 for 22 years at Oxford University, and then returning to industry at Microsoft from 1999 until present. While he was working in a university environment he could persevere in his research considering his objectives as long-term achievements. When he went back into industry, he found that assertions are in widespread use, and in a range of products comprise between one and ten percent of the code volume. Their primary role is to act as a test oracle, a definition of when and under what circumstances a test on that specific program is considered a failure.

According to him there are still a lot of challenges to face, like the extension of assertions to cover some characteristics of the object-oriented languages, such as inheritance, overriding and pointer manipulation. His belief however is that in the future assertional methods will be used as a design tool to evaluate the quality of programs. Keeping this aim in mind, it is still very important to concentrate on academic long-term research objectives.

Joe Stoy's talk was entitled "*The beginnings of formal semantics at Oxford,*" in which he described in detail the creation and the results of the Programming Research Group (PRG) at Oxford University. The group was the outcome of a strong battle between Leslie Fox and

Christopher Strachey who, at the beginning of the 1960s, had opposing views with regard to computing machines and the most appropriate use of them.

According to Strachey, programming demanded a great deal of mathematical and theoretical study, while Leslie Fox believed that it was mainly a practical activity that was not suitable for undergraduates. In fact Fox was very much against the practice adopted at MIT of using almost half of the available machines to teach students the programming principles and techniques. Strachey's major objective was the definition of the basic concepts that allowed the description of all the parts of a programming language in term of mathematical declarative expressions, so that it would not be necessary to postulate an "evaluating mechanism."

The contact between Strachey and Dana Scott was very fruitful both for themselves and the whole PRG. Dana Scott's work on lattice semantics allowed the use of typed  $\lambda$  calculus and, from 1969, of type free formal calculus. Strachey himself underlined Dana's role in his results, reporting progress directly to the Science Research Council (SRC) in 1970.

Stoy mentioned many PRG graduate students who made important contributions to research in the field, during the 1970s, but reported also that Strachey was seriously worried about the distance between programming practice in industry and programming theory studied at university. However, Stoy emphasized the increased importance of simplicity and of functional programming in industrial software production. Even if he started to work with the group by chance, being a physicist who happened to attend the right party at the right time, he has enjoyed being a member of the PRG.

Robin Milner is renowned for three distinct and complete achievements, which had a marked effect on the theory and practice of computer science: LCF the mechanization of Scott's logic of computable functions, probably the first theoretically based yet practical tool for machine-assisted proof construction; ML, the first language to include polymorphic type inference together with a type-safe exception-handling mechanism; and CCS a general theory of concurrency. He was the third speaker at the meeting and gave a talk with the title "*Concept and formality in computing.*"

He spoke about how his scientific life divided into four major interests: program verification, semantics, process algebra and models of interactions. The starting point was the necessity of testing large programs, and the desire to mechanize the program verification procedure. Having this purpose in mind, he created a resolution theorem prover that worked very well. This experience gave him the clear belief that he needed science and not luck! Creating interaction between man and machines implied use of the formality of the program structure in order to avoid misunderstandings. The machine assisted formal reasoning obliged the human programmer to express goals, proof strategies, and to define the notion of composing strategies together. He was influenced by Dana Scott and John McCarthy and spent one year at Stanford (1971-1972) working in the concurrency field.

In his view there is a balance between formal semantics and programming practice. He has belonged to different research communities and declared to have influenced and to have been influenced by all of his colleagues. He seemed to be very conscious of the dilemma between formality in languages and the need for quick and reasonable results in the actual practice of programming.

Peter Landin gave the last talk with the provocative title of "*Why are things so complicated?*" It was a very personal recollection of thoughts about the beginnings of his scholarly career, started at the end of the 1950s. He was much influenced by McCarthy and started to study LISP when the most common language was FORTRAN. LISP was very different from the other contemporary languages because it was based on a functional calculus rather than being procedural in nature. He reminded the audience of Marvin Minsky's hostility against  $\lambda$ -calculus and ALGOL, while he was writing some theoretical papers related to them. He remembered how difficult it was to deal with delay lines and drums and gave the flavor of the past times. The audience had the impression that a piece of the computing history was dancing in front of them.

At the end of the meeting, Cliff Jones, who was cited by some of the main speakers as one of the major scientists in the field, drew some conclusions. The ability to prove mathematically that a program correctly implements its specification is increasingly important, even if there is still a lot to do in order to guarantee that security and safety-critical applications perform correctly. The major points of importance were:

- The long-term objectives in research, that were not comparable with the urgency of short-

term results of software engineering companies;

- The results obtained through academic achievements used subsequently in industrial practice, confirming that academic ideas can be successful with patience;
- The never-ending tension between theory and practice in using formal methods;
- The importance of belonging to a scientific community in order to achieve outstanding results;
- The profitable interactions between some US universities and scientists such as Dana Scott and John McCarthy and UK research groups.

Further information on the meeting, including a selection photographs, can be found on-line:

<http://vmoc.museophile.sbu.ac.uk/pvs01/>

---

Published as: *Program Verification and Semantics: The early work*, Teresa Numerico and Jonathan P. Bowen. [IEEE Annals of the History of Computing](#), **24**(1):90-92, January-March 2002. In [Events and Sightings](#), Mary Croarken and Nathan Ensmenger, pp. 90-94.

Also in [BCS Computer Resurrection](#) (The Bulletin of the [Computer Conservation Society](#)), **27**:15-18, Spring 2002.