CV : Peter G. Hancock

Personal data

address	7 Cluny Avenue, Edinburgh, EH10 4RN
phones	(+44) 131-447-2555, (+44) 785-525-3381
email	**hancock@spamcop.net**
DOB, nationality	20 December 1951, British
degrees	Department of Computer Science, University of Edinburgh, 1996-2000 Doctor of Philosophy (Oct. 2000) Queens College Oxford, 1969-1972 Double Honours (2.1) Mathematics and Philosophy

Employment History

Nexwave Solutions, R&D Cambridge Jul 2002-Feb 2003	Senior Engineer Component based operating systems Last salary: £40,000 pa
Swansea University, Computer Science. Feb-July 2001	Fixed term Lecturer B. Developed and gave an Msc course, entitled 'Faults and Fault Tolerance'. Last salary: £24,227 pa
Digital, VMS Engineering. 1988-1995	Principal software engineer. Transaction processing, design of a queue manager, file system architecture, patent applications, formal specification, liason with universities. Last salary: £36,253 pa
Metier Management Systems Ltd. 1982-1988	Senior software engineer. Design of a message passing kernel and other system software for a database machine, board design, microcoding, diagnostics, system debugging.
Instron Ltd. 1981-1982	Software engineer. Signal processing, control engineering, systems programming.
Oxford University 1978-1981	Research assistant on psychology projects. Statistics, general programming, signal processing, systems programming.

Research, Publications

Edinburgh 1995-2000	PhD study, Laboratory for the Foundations of Computer Science, Thesis ‘Ordinals and Interactive Programs’ accepted October 2000: concerns interactive programming and proofs of well foundedness in constructive type theory. Peter Hancock, Anton Setzer: Interactive programs in dependent type theory. In: P. Clote, H. Schwichtenberg: Computer Science Logic. 14th international workshop, CSL 2000. Springer Lecture Notes in Computer Science, Vol. 1862, pp. 317 - 331, 2000.
Oxford 1985-1986	Wolfson Industrial Fellow, Programming Research Group Oxford University Computing Laboratory. (Secondment from Metier.) Functional programming: a self booting compiler. Chapters 8-9 of ‘The Implementation of Functional Programming Languages’, Simon L. Peyton-Jones, Prentice-Hall 1986.
Oxford,Uppsala 1972-1978	Research in mathematical logic. Type theory for constructive mathematics. Collaboration with Martin-Löf. “Hancock’s conjecture”. Visiting Lecturer 1976 Uppsala University Philosophy Department. Thesis abandoned 1978. Stockholm University Department of Mathematics preprint No. 3 1975, ‘Syntax and Semantics of the Language of Primitive Recursive Arithmetic’, with Martin-Löf.

Teaching

Edinburgh spring 2000	Tutorials and marking to support the second year computer science course.
Swansea Feb-July 2001	Msc course (CS-M1) ‘Faults and Fault Tolerance’, devised by myself.

Other Activities

Seminars on interactive programming at Heriot Watt, Nottingham, Swansea. Participated in workshop on formal topology, Gothenburg May 2003.

Participated in the Esprit Working Group APPSEM, giving the following talks

- "the Model of Computable Terms", at the Workshop on Normalization by Evaluation, Göteborg, 8-9 May 1998. (Abstract in BRICS-NS-98-1.)
- "Interactive Programs in Type Theory", at the Workshop on Subtyping and Dependent Types in Programming, Ponte de Lima, Portugal, 7 July 2000. (In Proceedings DTP'00.)

Opponent for Verónica Gaspes' PhD thesis at Chalmers in 1997, and for Henrik Persson's Licentiate thesis at Chalmers in 1996. External referee for Felix Joachimski's PhD thesis at Munich in 2001.

During Oct-Nov 2000 I visited the Mathematischen Institut der Ludwig-Maximilians-Universität München for 2 months, at the invitation of Prof. H. Schwichtenberg. The visit was funded by the Graduiertenkolleg Logik in der Informatik. Here I gave a short course on machine-checked well-ordering proofs, and various talks to the "Types club" in the computer science department.

In the last few years, I have refereed contributions to a number of journals and conference proceedings, and a review for Mathematics Reviews. Other reviewing is ongoing.

Research Interests

I have worked as an engineer in real-time programming, low-level systems design, and on large operating systems projects connected with transaction processing. There is a tension between my experience as an engineer and my interests in logic.

My interests include the following.

- Models of interactive behaviour in constructive type theory.
  Together with Anton Setzer, and Pierre Hyvernat I have worked on models for interactive programs in dependent type theory. The approach is aligned with the so-called 'IO monad' used in functional programming languages such as Haskell. We published a paper about this in CSL 2000. Some experimental implementations was carried out by Pierre Hyvernat in 2001.
  I have worked out a model of command-response interfaces and components, that captures a significant part of an important ingredient of one notion of system component with some demonstrable practical interest. This model provides a constructive reading of Back and von Wright's refinement calculus. There are close connections with 'formal topology' as developed by Sambin, according to which a system is a weak counterpart of a topological point.
  Together with Anton Setzer, I have explored one approach to a foundation for coinduction in constructive type theory. The approach is based on a weak form of the second-order existential quantifier. This work was presented at a meeting in Dagstuhl in 2001.
- I've had a long standing interest in mathematical logic, particularly ordinal-theoretic proof theory. This subject concerns with the extent to which a formal system can recognise well-foundedness of relations, termination of computations, and related phenomena. The 'extent' can be measured by a countable ordinal. My PhD thesis included a machine checked proof of part of an old conjecture of mine concerning the ordinal of a certain weak form of Martin-Löf's type theory. The proof is essentially an algebraic study of a certain kind of non-monotone predicate transformer, deeply involved in Gentzen's proof. It may be that this notion can be applied to some stronger forms of type theory, simplifying existing work in proof theory.
- Having had 'software engineer' in my job title for about 15 years, I am professionally interested in what good engineering practice is, in the mix of skills, talents and qualities one needs to turn ideas into on-time successful products. I'm not sure that software engineering amounts to a systematic body of teachable objective knowledge. I am interested in what it means to apply mathematics, particularly in engineering. Perhaps these are not formally research interests.

Referees

(Not posted on web)