CURRICULUM VITAE
(revised in January 2008)

Name Arthur John Robin Gorell Milner

Date and place of birth 13.1.34, Yealmpton

Family Married with two grown-up children.

Nationality UK

Appointment Emeritus Professor, Computer Science, University of Cambridge.

Office address The Computer Laboratory, University of Cambridge, J J Thomson Avenue, Cambridge CB3 0FD, UK. Phone: +44 1223 334718.

Home address 24 Lyndewode Road, Cambridge CB1 2HN, UK.

University Degrees and Honours

- Cambridge 1957: B.A. (Maths Part II Class 1 1956, Moral Sciences Part II Class 2.1 1958).
- Chalmers University, Gothenburg, 1988: Honorary Doctorate.
- University of Stirling 1996: Honorary Doctorate.
- University of Bologna 1997: Honorary Doctorate.
- City University London, 1998: Honorary Doctorate.
- Aarhus University, Denmark, 1999: Honorary Doctorate.
- University of Essex, 2000: Honorary Doctorate.
- University of Edinburgh, 2003: Honorary Doctorate.
- University of Swansea, 2004: Honorary Fellowship.
- University of Glasgow, 2005: Honorary Doctorate.
- University of Paris Sud, 2007: Honorary Doctorate.

Awards and distinctions

- British Computer Society Technical Award, 1987 for “The Programming Language ML”, a 12-year project of which I was team leader.
- Royal Society of London, 1988: Fellow.
- Academia Europaea, 1988: Founder Member.
- British Computer Society, 1988: Distinguished Fellow.
- ACM A.M. Turing Award, 1991.
- Royal Society of Edinburgh, 1993: Fellow.
- ACM 1994: Fellow.
- ITALGAS prize, Turin, 1994.
- F.L. Bauer prize, Munich, 1994.
- ACM SIGPLAN award for Achievement in Programming Languages, 2001.
- UK Computing Research Committee, 2001.

– Royal Medal, Royal Society of Edinburgh, 2004
– Distinguished Achievements Award, European Association for Theoretical Computer Science, 2005.
– Foreign member, Académie Francaise des Sciences, 2005.
– Foreign associate, National Acaedmy of Engineering, USA, 2008.

Career since Graduation

– 1952–54, Military Service, Royal Engineers (2nd Lieutenant).
– 1959–60, Marylebone Grammar School, London, Mathematics Teacher.
– 1960–63, Ferranti Ltd, London, Computer programming and related work.
– 1963–68, The City University, London, Lecturer in Mathematics and Computer Science.
– 1968–71, University College, Swansea, Glamorgan, Senior Research Assistant (Computer and Logic Group).
– 1971–73, Stanford University, California, USA, Research Associate at Artificial Intelligence Project, in Computer Science Department. (Math.Theory of Computation).
– 1973, Edinburgh University, appointed lecturer in Computer Science.
– 1975, Edinburgh University, Senior lecturer.
– 1979, Edinburgh University, Reader.
– 1979–80, Aarhus University, Denmark, Guest Professor in Computer Science (6 months).
– 1984, Edinburgh University, Professor.
– 1986–89, Edinburgh University, Director of Laboratory for Foundations of Computer Science.
– 1990–94, Edinburgh University, SERC Senior Research Fellow.
– 1995– , University of Cambridge, Professor of Computer Science (Head of Department from January 1996 to September 1999).
– 1999–2001, University of Cambridge, Research Professor.
– 2001– , University of Cambridge, Emeritus Professor.
– 2006–2007, Île de France, Blaise Pascale Chaire internationale de recherch.

Research interests Mathematical Theory of Computation; in particular:

– Formal deductive systems for proofs about mathematics and computation, and the methodology of conducting such proofs with computer assistance.
– The design and formal definition of programming languages.
– Abstract models of computation, especially of concurrent interactive systems; their algebraic and logical theories and their use in language design, semantics and system analysis.
– Models of mobile informatic processes, from business to biology.
– A strategy for developing pervasive computing under scientific control.

Invited lectures These lectures are at conferences, colloquia, advanced courses etc. (One lecture, unless otherwise indicated).

– International Symposium on Theoretical Programming, Novosibirsk, USSR, 1972.
– Conference on Informatics Theory, Pisa, Italy, 1973.
– Advanced Course on Foundations of Computer Science, Amsterdam, 1976 (6 lectures).
– IBM Symposium on Mathematical Foundations of Computer Science, Amagi, Japan, 1976.

- AFCET/SMF Joint Colloquium in Applied Mathematics, Paris, 1978.
- 7th International Symposium on Foundations of Computer Science, Poland, 1978.
- 8th ditto, Czechoslovakia 1979.
- 4th G.I. Conference on Theoretical Computer Science, Aachen, Germany, 1979.
- Inter-Universities Computing Colloquium, Exeter, 1980.
- SRC conference on Theoretical Aspects of Distributed Computing, Loughborough, 1980.
- French Summer School on Petri nets and Parallel Processing, Colville-sur-mer, May 1980.
- 6th CAAD Colloquium (Trees and Algebra in Programming), Genoa, 1981.
- Workshop on Foundations of Software Technology, Bangalore, India, December 1981 (6 lectures).
- British Computer Society, Formal Aspects of Computer Science workshop on the Calculus of Communicating Systems, July 1982 (2 day lecture course on the subject of my book).
- IEEE Symposium on Information Theory, Quebec, September 1983.
- IFIP Congress, Paris, Sept 1983 (Chairman of Panel Discussion).
- Royal Society Discussion meeting on "Mathematical Logic and Programming Languages", London, February 1984.
- Massachusetts Institute of Technology, Computer Science Dept, Distinguished Lecturer, February 1984.
- University of Chicago, Guest Professor for 1 week, February 1984, (3 lectures).
- International Summer School on "Control Flow and Data Flow: Concepts of Distributed Programming", Marktoberdorf, August 1984 (6 lectures).
- SERC Conference on Distributed Computing, University of Sussex, September 1984. Pre-conference tutorial (3 lectures).
- International Conference on Petri Nets, Finland, June 1985.
- 6th Congress of Informatics and Automatic Control, Madrid, October 1985.
- Workshop on Design Methodology for Distributed Systems and VLSI,
- Swedish Institute of Computer Science, Kista, Sweden, January 1986.
- IFIP Congress, Dublin, July 1986.
- International Congress of Mathematicians, Berkeley, California, August 1986 (invited speaker, but unable to attend).
- TAPSOFT, Joint Conference on Theory and Practice of Software Development, Pisa, March 1987.
- London Mathematical Society, Workshop for mathematicians on Theoretical Computer Science, Univerity of Sussex, May 1987.
- Logics in Computer Science, International conference, Cornell University, USA, June 1987.
- Computing meeting for Mathematicians, London Mathematical Society, Sussex, May 1987.
- 4th British Colloquium for Theoretical Computer Science, Edinburgh, March 1988.
- Combining Compositionality with Concurrency, Workshop at GMD Bonn, March 1988. "Unique Decomposition of Processes".
- Conference on Mathematical Structures for Software Engineering, IMA, at Manchester Polytechnic, July 1988.
- International Conf. on Parallel Computation, CONPAR 88, Manchester, September 1988. 1-day tutorial: "Formal Methods for Concurrency".
- International Conference on Fifth Generation Computing Systems, Japan, December 1988. "Interpreting one concurrent calculus in another".

– TAPSOFT, Brighton, April 1991. “Movement among Processes”.

– UK Association for Logic Programming Conference, Edinburgh, 1991. “The π-calculus: a model for moving objects”.

– Colloquium for 65th birthday of Carl Adam Petri, Univ of Hamburg, 1991. “Automata, Algebra and Concurrency”.

– International Summer School on “Logic and Algebra of Specification”, Marktoberdorf, July 1991 (4 lectures).

– VMD ’91 Conference, 21–25 October, 1991. “The π-calculus: a model for moving objects”.

– 1991 Turing Award Lecture: ACM National Conference, Kansas City, USA, February 1992.“Elements of interaction”.

– JFIT Technical Conference, Keele University, 1993. “Application of theory, or theory from applications?”.

– International Summer School on “Proof and Computation”, Marktoberdorf, July 1993 (5 lectures).

– Conference on Foundations of Computing Theory, Szeged, Hungary, August 1993. “An action structure for the synchronous π-calculus”.

– Conference on Math. Foundations of Computer Science, Gdansk, Poland, August 1993. “Action calculi, or syntactic action structures”.

– Computer Science Logic, University College Swansea, Sept 1993. “Higher-order action structures”.

– TAPSOFT, Edinburgh University, 1994.

– 13th World Computer Congress, IFIP 1994, “Computing is interaction”.

– The Geary Lecture, City University (London), May 1996. “Computing is interaction”.

– ACM Federated Computing Research Conference, Philadelphia, May 1996. Keynote lecture: “Computing is interaction”

– 11th Annual IEEE Symposium on Logic in Computer Science, part of the Federated Logic Conference, Rutgers University, New Jersey, July 1996. Plenary Lecture for the Federated Conference: “Calculi for Interaction”.

– IEE International Workshop on Discrete Event Systems (WODES), Edinburgh, September 1996. Keynote lecture: “Pi Calculus and its Applications”.

– The J Barkley Rosser Memorial Lecture, University of Wisconsin, Madison, Wisconsin, April 1997.

– International Conference on Automata, Languages and Programming (ICALP), Bologna, July 1997.

– Conference on Theory and Applications in Computer Science (TACS), Japan, September 1997.

– Conference on Mathematical Foundations of Programming Semantics, Hoboken, New York, April 2000: “Labelled transitions for graphical reactive systems”.

– The Saul Gorn Memorial Lecture, University of Pennsylvania, May 2001: “How to model mobile computing”.

– 22nd International Conf. on Applications and Theory of Petri nets, Conference on Petri Nets, Newcastle, UK, June 2001: “The flux of interaction”.

– 12th International Conference on Concurrency Theory (CONCUR 2001), Aalborg, Denmark, August 2001: “Bigraphical reactive systems”.

– British Conference on Theoretical Computer Science, Bristol, April 2002: “Bigraphical reactive systems”.

– SIGPLAN 2002 Conference on Programming Language Design and Implementation (PLDI), Berlin, June 2002: “Languages, models and the global machine”.

– First International Conference on Graph Transformation, Barcelona, October 2002: “Bigraphs as a model for mobile interaction”.

– Summer School on Petri Nets, Eichstätt, Germany, September 2003: Advanced course (3 lectures) on “Bigraphs and mobile processes”.
– European Joint Conference on Theory and Practice of Software (ETAPS), Barcelona, April 2004: “The global ubiquitous computer”.
– World Computer Congress, Toulouse, August 2004: “A scientific horizon for computing”.
– Conference on Category Theory in Computer Science, Copenhagen, August 2004: “Structural mathematics for mobile computation”.
– Sixth International Conference on Ubiquitous Computing, Nottingham, September 2004: “Science for global ubiquitous computing”.
– Conference on Converging Sciences, Trento, December 2004: “Scientific foundation for global computing”.
– European Science Foundation ‘forward look’ meeting on Nano Science and Information Technology, April 2005: “Anarchical computing”.
– Institut National de Recherche en Informatique et Automatique (INRIA), conference to celebrate the 40th anniversary of the Laboratory, December 2007: “Ubiquitous Computing: Shall we understand it?”.

Examining

External, undergraduate: 1995–97 University of Warwick

Internal, undergraduate: 1996 University of Cambridge

External, PhD: 1974 Warwick (Hitchcock), Essex (Seddon); 1975 Oxford (Milne); 1977 London (Miyahara), Oxford (Chris Jones), Paris (Greussay); 1979 Paris (Berry); 1980 Aarhus (K. Jensen); 1981 Oxford (Cliff Jones); 1983 Oxford (Brookes); 1987 St Andrews (Livesey); 1991 Imperial College (Thomsen); 1991 Sussex (Aceto); 1993 Geneva (Dami); 1995 Imperial College (Gay); 2003 Paris (Fournet); 2006 Paris (Hym); 2006 Paris (Krivine).

Internal, PhD: (at Stanford) 1971-72 J.M. Cadiou, J. Vuillemin, L. Morris.
(at Edinburgh) 1973-83 J. Moore, M. Gordon, R. Topor, G. Lev, G. Winskel, L. Cardelli, G. Brebner; 1984 Hanne Riis Fleming; 1985 R. de Nicola; 1989 E. Moggi; 1991 Sun Yong; 1991 J. Bradfield, H. Hüttel.
(at Cambridge) 1995 I. Stark; 1996 A. Kennedy, F. Davey, M. Nesi; 2000 P. Wojciechowski.

PhD Research Students

1975-1983 R. Aubin, A. Cohn, G. Milne, A. Mycroft, M. Sanderson; 1984 L.Damas; 1985 B. Monahan, K. Larsen; 1986 K. Mitchell; 1988 K.V.S. Prasad, M. Tofte; 1989 F. Moller; 1990 D. Berry, C. Tofts; 1993 D. Sangiorgi; 1995 P.E. Sewell; 1996 D.N. Turner, A. Mifsud; 2001 J. Leifer; 2006 O.-H. Jensen.

Conference programme committees

– Program Proving and Improving, Arc-et-Senans, 1975.
– 3rd International Colloquium on Automata, Languages and Programming, Edinburgh, July 1976 (organizing member).
– IFIP Working Conference on Formal Description of Programming Concepts, St. Andrews, Canada, August 1977.
– International Conference on Mathematical Studies of Information Processing, Kyoto, Japan, August 1978.
– 7th International Symposium on Mathematical Foundations of Computer Science, Poland, 1978.

– (co-chairman) Conference on Semantics of Concurrent Computation, Evian-les-Bains, France, July 1979.
– 2nd International Symposium on Automated Deduction, France, 1980.
– 7th International Colloquium on Automata, Languages and Programming, Amsterdam, 1980.
– 8th Ditto, Israel, 1981.
– 9th Ditto, Denmark, 1982.
– International Colloquium on Formalization of Programming Concepts, Peniscola, Spain, 1981.
– International Conference on Very Large Scale Integration, Edinburgh, 1981.
– 2nd International Symposium on Programming, Toulouse, April 1984.
– 4th International Conference on Automated Deduction, July 1984.
– European Symposium on Programming, Saarbrucken, March 1986.
– 7th European Workshop on Application and Theory of Petri Nets, Oxford, July 1986.
– IFIP TC2 (Theory of Computation) Conference, August 1986.
– TAPSOFT, Pisa, March 1987.
– Conference on Functional Programming and Computer Architecture, 1991.
– CONCUR, April 2000.

Public service

– Past council member of the European Association for Computer Science Theory.
– Past editor for Journal Theoretical Computer Science; current editor for Research Notes in Theoretical Computer Science (a monograph series), the Journals of Formal Aspects of Computer Science, Mathematical Structures in Computer Science, Formal Methods of System Design, and the Computer Journal.
– 1981–84: Member of the SERC Distributed Computing panel.
– 1990–93: Founding chairman of the UK Distinguished Dissertations Scheme in computer science, sponsored by the Conference of Professors of Computer Science and the British Computer Society. I designed the scheme; it runs now almost unchanged.
– 1988–89: Member of Mathematical Sciences Subcommittee, University Grants Commission.
– 1988–91: Member of Information Technology Advisory Board (ITAB).
– 1989: Member of research assessment panel for Computer Science, Universities' Funding Council.
– 1990: Member of teaching assessment panel for Computer Science, Universities' Funding Council.
– 1991: Coordinated submission from leading UK computer scientists to SERC Review of Information Technology; "Computer Science: the Core IT Research Discipline".
– 1992: Member of research assessment (RAE) panel for Computer Science, Universities' Funding Council.
– 1993– : Editorial board, Royal Society of Edinburgh Journal of Mathematics.
– 1993: External assessor for undergraduate courses in Computer Science, at Imperial College London.
– 1996: Chairman of research assessment (RAE) panel for Computer Science, joint Higher Education Funding Councils.
– 1996: External Assessor for Danish National Research Foundation, on establishment of PhD schools.
– 1997: Chairman of External Review Panel on future plans for the Informatics Planning Unit, Edinburgh University.
– 2001–2: Chair of Sectional Committee 1 of the Royal Society, responsible for elections in mathematics and computer science.

– 2002– : UK Computing Research Committee (founding member).

– 2002– : Foresight Panel, IT University of Copenhagen.

Teaching activity I first taught computation, in the form of formal languages and automata theory, at City University in 1963–68.

On arriving in Edinburgh in 1973, I took a leading part in bringing theoretical content into the teaching of Computer Science. At that time it was almost completely absent from Computer Science syllabuses, both in Edinburgh and elsewhere. Since I was the first permanent staff member with mainly theoretical interests to be appointed in the department, it fell to me to propose appropriate theoretical courses. By 1990 about a third of the full-time teaching staff, and a higher proportion of the research staff, had active theoretic interests. The undergraduate course in Edinburgh is now one of the broadest, both theoretically and otherwise, in the country.

In October 1977, with R.M. Burstall, I initiated a postgraduate course computing theory for first year PhD students, with the aims of (i) attracting qualified mathematicians into computing research – and eventually teaching at all levels – in UK, and (ii) allowing graduates in computing, who have kept up their mathematics, to integrate the two with some purpose. Some five or six new PhD students took part each year, including many from abroad and many mathematicians. The experiment worked; Edinburgh's contribution to academic Computer Science through these PhD graduates is now well recognised both in UK and abroad. Some, such as Luca Cardelli and Mark Jerrum, are now internationally renowned.

I continually tried to make the theoretical and practical threads of the undergraduate syllabus more relevant to one another. Although I taught mostly theoretical topics, I ran the general first-year Computer Science course for two years, 1982–1984, and tried to inculcate an appreciation of rigour. Later I started two new honours courses which bridge the theory/practice gap: Communication and Concurrency, and Language Semantics and Implementation.

Indirectly, the design of the programming language Standard ML (which I led) has had a considerable effect on teaching, since it is taught to undergraduates thoroughout the world – often, for example at Cambridge, as the first language. My textbook "Communication and Concurrency" is also taught at a higher level.

In April–May 1993 I convened a working party on the degrees offered by the Informatics Planning Unit at Edinburgh, which comprises the Computer Science Department, the Artificial Intelligence Department and the Centre for Cognitive Science. The remit of the Working party was to examine all taught degrees offered, and identify ways to develop and to integrate them.

On arrival in Cambridge in January 1995 I resumed teaching. For five years I taught a course in Communicating Automata and the Pi Calculus, based upon research by myself and colleagues. I also lectured on Designing and Defining a Programming Language, based upon my experience with Standard ML, and gave single introductory or perspective lectures in my role as Head of Department.

Research activity My research in Computer Science has been always to do with foundations. The seeds were sown during my period at City University in 1963–1968; while there, I formed a deep interest in automata theory, programming languages, artificial intelligence and the relationship of logic with computation. I did not publish anything significant then, but these interests have remained with me ever since. My original contribution began when I took a research post at Swansea in 1968; there I became concerned with rigorous methods for analysing computer programs.

This concern became more specific at Stanford in 1971, where I began to ask how the error-prone business of program verification could be made more robust with computer assistance. This —and the influence of Dana Scott— led me to develop LCF, a Logic for Computable Functions, in which propositions about algorithms, programming languages and computing systems could be precisely formulated; a computer program based upon this logic allowed one to interact with the machine in conducting proofs. This line of research still continues; it has become more markedly a method of expressing proof *strategies*, both complete and partial, in a structured way. With this mode of expression it becomes easier to transfer the tedium of proof —and even some of its clever parts— to the computer, but still retain the vital element of human participation.

First at Stanford, then at Edinburgh, later at Cambridge, Cornell, Gotheburg and Paris, proof systems using many of these ideas have bridged the gap to practical application. For example, Michael Gordon at Cambridge has used many of the ideas and some of the software in his HOL system, a mechanized higher-order logic in which he and his team have had considerable success in verification of hardware systems; this has led to a new industrial methodology in computer design. Also Robert Constable and his Cornell group have successfully applied their NuPRL system, derived partly from LCF but directed to constructive proof, to large problems, some previously posed but unsolved in mathematics.

A spin-off from the LCF work is a general-purpose programming language called ML, which stands for "Meta-Language" (since it was the language for driving the LCF proof system). This language is now taught to undergraduates around the world, and has been used in producing the first computer-checked verifications of hardware systems in UK, at RSRE Malvern and at INMOS. ML embodies many ideas already known to the Artificial Intelligence community in 1974 and earlier; its main and crucial innovation is a new rigorous but flexible type discipline. Indeed, this type discipline was almost *forced* into existence by the demand for complete rigour in machine-assisted proof on the one hand, and the need for flexibility in performing these heavy proofs on the other hand. Despite this specialized motivation, its benefit to programming appears to be very general.

The ML language was adopted by the Alvey programme; it also won the Technical Award of the British Computer Society in 1987. In 1989 it evolved into Standard ML, with parametric modules for programming-in-the-large and a published fully formal semantic definition. I led this evolution, which involved a team of designers and critics around the world for some years. Seven years later, with the other main designers, I reviewed the language and published a simplified formal Definition that incorporates several improvements arising from the first seven years' experience with the language. The ML type discipline, if it had been adopted commercially even two decades after it was defined, could have prevented the furious and hugely expensive uncertainty surrounding the Y2K problem; the problem was not that things went wrong, but that everyone feared that they would and no-one could predict that they wouldn't!

The main strand of my work is the attempt to understand interacting concurrent processes from a mathematical viewpoint. This also began at Stanford, in 1972. Increasingly since then I have felt that concurrent processes need a new kind of theory, that existing attempts are inadequate, and that to construct it presents one of the main challenges in Computer Science. A mark of success will be when designers of concurrent systems begin to use a coherent body of *concepts and mathematics*, not just a variety of formalisms. My 1980 book "A Calculus of Communicating Systems" (CCS) aimed in this direction, and gained considerable recognition. CCS presented many conceptual problems in a precise form, and solved some of them. The theory, considerably extended and also streamlined, was reported in a new book "Communication and Concurrency" in 1989. CCS has been applied widely in industry, and was used in the foundation of LOTOS, a communications description language which is an ISO standard. These applications —besides justifying the effort— are very useful in feeding back reinforcement or criticism of the theory.

A weakness of CCS and all of its rivals was that it could not model *mobility* — the ability of processes to change their interconnections dynamically. This was not an oversight in CCS; I just could not see how to do it properly. But it the late 1980s, with colleagues both at Aarhus and at Edinburgh, I developed it into the π-calculus, which not only caters for arbitrary reconfiguration within systems, but also —unlike CCS— admits a general treatment of data, subsumes functional programming, and even provides many of the primitive ideas for object-oriented programming. It seems a candidate for a fundamental calculus of concurrent computation. Much recent research of my group at Edinburgh in the early 1990s was devoted to developing it for this purpose. A significant step was to show that it naturally subsumes the λ-calculus (the canonical functional calculus). Recently (2002) the π-calculus has been widely adopted as a model underlying software for business processes.

Mobile computing systems —whether they move in virtual or physical space— are increasingly important. From 1991 to 1995, on an EPSRC Senior Research Fellowship at Edinburgh, I was concerned with a mathematical *framework* for mobile calculi such as the π-calculus. The point is this: to establish the status of a calculus it is not enough just to apply it to this and that, nor just a matter of seeing whether its mathematics works out properly, though these are necessary tests. More difficult is the question of whether it, or some rival calculus, can claim to be canonical. To compare candidates ones needs a framework; this led me to invent *action calculi*, leading in 2001 to *bigraphical reactive systems*. A general study of this kind always runs the risk of bland vacuity, but present progress seems to justify the broader attack. For

example, with students I have recently found that the dynamic theory of a wide range of calculi, including the π-calculus, can be recovered uniformly in the bigraphical framework.

Applications of these models are broadening. Through the work of colleagues at the Weizmann Institute (Israel) and Microsoft (Cambridge), we begin to model biological processes; biologists are actively exploring computational models. At the University of Trento (Italy) a large centre for this work has been recently established with funding from Bill Gates. Collaboration with the IT University at Copenhagen will lead to the practical specification and implementation of sentient environments, an essential part of ubiquitous computing. Finally, companies are employing these models for business processes (transactions, mergers, contracts, ...), and the Worldwide Web committee is seeking to use them to establish standards.

I continue to do detailed technical research on the kinds of mathematical model that can underpin ubiquitous computation, discussed below. In this line, in 2009 Cambridge University Press will publish my book *The Space and Motion of Communicating Agents.*

Organisation of research From 1974, continuously until the end of the century, I held several government grants for research projects. In 1986, when the Alvey Directorate was initiated, my colleagues and I were concerned that the drive for industrial exploitation of research —worthy in itself— might starve basic research in favour of short-term projects. This balance has constantly to be fought for. At that time we were able to attract sufficient funds from both SERC and the Alvey Directorate to found the Laboratory for Foundations of Computer Science (LFCS) at Edinburgh. LFCS expanded later to some 70 members. It remains firmly committed to foundational research —since Computer Science badly needs it, and only Universities look like supplying it— but it also has interaction with industrial applications. I was the founding Director of LFCS. It became known worldwide as a leader of research in theories of computing.

Reinforced by my experience with theoretical research, I am committed to the idea that Computer Science will be a science as fundamental as Physics, and with the same power to change and be changed by mathematics. For a while it was just the study of what goes on in a computer, and if that were all then such a claim would be too bold. But distributed computing, communications, "parallelism" and many interactive applications have not only broadened that study; they have shown us how to lift it to a higher plane. As mentioned above, biological processes are also modelled. It is better now to talk of "Informatics" than of "Computer Science", since we are dealing with the understanding of systems of all kinds —not just man-made— through the concept of information flow. The algorithms and machines with which Computer Science began some four decades ago are just a small part of this much larger story.

A motivation for my coming to Cambridge in 1995 was to work near a larger group of practical researchers, in order to find out which abstract models are most relevant to the design of real systems. This has been a great help in research. On the one hand I am greatly encouraged that theories of mobile processes are highly relevant to the modern interactive computing world; on the other hand, I learn from practical colleagues what issues are most important to them, and this steers the work.

A most recent involvement has been as joint leader, with C.A.R. Hoare, of a UK movement to establish long-term research goals in computing, including identification of Grand Challenge projects. A particular Grand Challenge we have posed is entitled *Ubiquitous Computing: Experience, Design and Science*; it recognises that to understand the informatic systems of the 21st century will entail a close-knit collaboration among researchers at three levels: the interface between humans and ubiquitous systems, the establishment of design principles for ubiquitous systems (which will often serve us without explicitly communicating with us), and the creation of an underlying analytical theory for such systems. I am deeply involved in the theoretical component; in particular, we aim to ensure, within two decades, that the design of every large software system should be expressed not in an adhoc programming language but in the terms of a scientific informatic model —or rather a hierarchy of models— that treat man-made and naturally occurring systems equally. I argue that this need is more pressing than ever, given the ubiquitous and life-pervading nature of modern computing. If we do not attain scientific understanding (which, after all, underpins every engineering discipline *except* software!) the complex and adaptive nature of ubiquitous software will lead to its stagnation, or even to disaster. If we do attain it, then the opportunities are unlimited.