designed by unsual resources, the university of edinburgh

Further information from:

The Administrative Officer

Department of Computer Science,

James Clerk Maxwell Building,

The King's Buildings, Mayfield Road,

Edinburgh EH9 3JZ, UK.

Telephone: 031 650 5112

Facsimile: 031 667 7209

Departmental Report 1992-93

DEPARTMENT OF COMPUTER SCIENCE

Computer Science
Departmental Report

1992 - 1993

Contents

Departmental Computing Facilities

The computing resources of the Department of Computer Science and LFCS comprise mainly Unix workstations and support servers, networked together *via* Ethernet and FDDI local area networks. Sun SPARC workstations running Sun's version of Unix BSD 4.3 predominate, though there is a small number of HP and DEC Unix machines.

The 160 or so workstations are supplemented by 90 X and 140 RS232 terminals. Computation is done either on desktop machines or on one of the dozen or so compute servers ranging in size from the 70+ MIP LFCS Sun 4/600 down to the 25 MIP Sun 4/75s. Of these some 35 workstations, 60 X terminals and six computer servers are dedicated to student teaching.

The Department also supports some 50 or 60 Apple Macintoshes, a dozen PCs (mainly used for student ECAD).

All machines (Macintoshes and PCs included) are attached to the Department's Ethernet. The Macs can access the Ether through Appletalk/Ethernet gateways. the PCs have direct PC-NFS Ethernet connections. All can fileserve from the Unix mainframes and have provide access to a variety of special purpose peripherals including high quality laser document printers and plotters as well as line printers. The Department has always maintained a high degree of consistency across its network and has recently produced conference papers describing the innovative work in this area.

A rich variety of software is available at Edinburgh and visitors from well-appointed CS laboratories around the world find the environment very familiar. As well as standard network services (e.g. Internet FTP), GNU and LaTeX products, we offer AutoCAD, Maple, numerous programming languages, Oracle and Ingres databases, spreadsheets and presentation graphics packages. Additional material is constantly being retrieved from the Internet. Electronic mail and USENET news are integral parts of the environment and are extensively used.

The Department of Computer Science and Edinburgh University Computer Services (EUCS) were jointly involved in the setting up of a microcomputer laboratory in Appleton Tower at the central University site. This comprises mostly Apple Macintosh and PC clones along with a selection of terminals, linked *via* Ednet, accessing the EUCS machines (principally two Sequent Symmetry Unix and a Vax 8600 using VMS). For more compute-intensive problems, Supercomputer time may be negotiated with the Edinburgh Parallel Computing Centre (EPCC), which offers a range of services on its machines which include a Connection Machine CM-200 and Transputer and i860 arrays.

Ph.D. Graduates

P. Goldberg: *PAC-Learning Geometrical Figures*

K. Goossens: *Embedding Hardware Description Languages in Proof Systems*

C. Hermida: *Fibrations, Logical Predicates and Related Topics*

T. Hopkins: *The Design of a Sparse Vector Processor*

C. Liang: *Timed Processes: Models, Axioms and Decidability*

S. Manoharan: *Task Assignment in Parallel Systems*

J. McKinna: *Deliverables: A Categorical Approach to Program Development in Type Theory*

M. Mendler: *A Modal Logic for handling Behavioural Constraints in Formal harware Verification*

D. Sangiorgi: *Expressing Mobility in Process Algebra: the First-Order Versus the Higher-Order Pardigm*

F. Da Silva: *Correctness Proofs of Compilers and Debuggers: An Approach Based on Structural Operational Semantics*

G. Wilson: *Structuring and Supporting Programs on Parallel Computers*

P. Yeung: *High-Level Synthesis of VLSI Circuits*

The Department

The University of Edinburgh is beginning to recover from a difficult financial situation which has resulted in considerable loss of staff as well as increased student numbers. From being well funded as a consequence of expansion under the UK University Funding Council's Engineering and Technology Programme, the Department finds itself under considerable strain in terms of personnel and capital resources. At the same time, in common with all university departments, we are faced with an ever increasing administrative burden as we produce mountains of paper for externally imposed exercises such as teaching quality audit, teaching quality assessment and research assessment. Despite this, staff in the Department continue to strive for excellence in both teaching and research, with confidence boosted by the accreditation of our main single and joint courses as appropriate for C.Eng status by the BCS in March 1993, and by our 5A rating (the highest attainable) in the UFC's 1992 Research Assessment Exercise.

The main change in our teaching this year has been the introduction of a revised structure in first year, allowing some streaming of students, into two groups, those with previous knowledge and experience, and those without. Despite some teething troubles this year this move has been very successful and gives a strong base for future course developments.

In research an important new development within the LFCS is the expansion of work on safety critical systems. Members of LFCS are leading a recently funded interdisciplinary project which involves the Human Communications Research Centre, the Departments of Artificial Intelligence and Sociology, and a number of major industrial companies. In addition, a Teaching Company Scheme has been set up to effect technology transfer of related work in this area, itself a welcome new departure for the Department.

The Department is pleased to congratulate Professors Robin Milner and Gordon Plotkin on their election to Fellowships of the Royal Society of Edinburgh and to congratulate Professor Plotkin and Dr Don Sannella on their SERC Fellowship awards, tenable from October 1992. We welcome as their temporary replacements Dr Julian Bradfield, whose research interests are in modal logic and computer-assisted verification techniques, and Dr Todd Heywood.

Dr Heywood's interests are in parallel computing systems, particularly models of parallel computation. His arrival has been catalytic in bringing together several members of the Computer Systems Group to work on a SERC-funded project which will investigate parallel architectures to support these theoretical models, especially Todd's own Hierarchical PRAM model.

Professor R.N. Ibbett, Head of Department
August 1993

Introduction

The City and its University The University of Edinburgh was founded over 400 years ago in 1583, some twenty years before the Union of the Crowns which made James VI (I) king of both Scotland and England. Throughout its history it has numbered a host of distinguished scholars amongst its teachers and students. Its situation in the capital of Scotland, one of the world's most beautiful cities, has helped maintain this tradition.

Today, the University of Edinburgh is one of the largest in the UK. Its students enjoy not only the advantages of studying at a well-endowed university but also the many cultural, sporting and social aspects of the city. One advantage of studying at Edinburgh is the unusually large number of optional courses which the University, because of its size, can offer to its students and the resulting flexibility of its degree system.

The Department of Computer Science The Department of Computer Science at Edinburgh was established with six staff in 1966 when the functions of the Computer Unit (established in 1963) were divided between teaching (in the Department) and service (in the Edinburgh Regional Computing Centre, now the Edinburgh University Computing Service). The Department has since grown into one of the largest departments in the Science and Engineering Faculty with around 26 teaching staff supported by a similar number of computing officers, technicians and secretaries.

At undergraduate level the department offers a four-year Honours degree in Computer Science together with joint Honours degrees with Artificial Intelligence, Electronics, Management Science, Mathematics, Physics, and Statistics. There is also an alternative first year course in Information Systems aimed at students in other Faculties.

At postgraduate level there is a flourishing community of research students working towards the degrees of Ph.D. The department also runs an M.Sc. in Computer Science with three themes: Computer Systems Engineering, Parallel Systems, and Theoretical Computer Science.

Research in the Department of Computer Science started in 1966 with a multi-access project from which developed the interactive computing system EMAS (Edinburgh Multi Access System), which was used for some twenty years on a succession of mainframe computers to provide the University's main computing service. Since then the Department has gone on to gain an international reputation for the quality of its research, receiving top rating in both the 1989 and 1992 UFC Research Selectivity Exercises. There are several very active groups working in a number of different fields including: computational complexity, parallel systems, theory of computation and VLSI design using both formal and informal methods.

The Department is involved with two major research units within the University. The Laboratory for the Foundations of Computer Science is a federation

Claudio Russo *D.T. Sannella, K. Mitchell*	The design of module systems for typed, functional programming languages.
Roger Sayle *M.P. Fourman, R. Milner*	Use of formal methods, in particular higher order logic in the design of VLSI circuits.
Thomas Schreiber *R. Burstall*	Verifying imperative programs using Lego.
Dilip Sequeira *M.P. Fourman*	Semantics and language design.
Peter Sewell *R. Milner*	Divergence and abstraction in process algebra.
Alex Simpson *G.D. Plotkin, D. Pym*	Type theory and logical frameworks.
Colin Smart *D.K. Arvind*	Parallel simulation; distributed algorithms.
Makoto Takeyama *R. Burstall, Z. Luo*	Type theory and category theory with a particular interest in fibred categories, applications to proof.
Hayo Thielecke *S.O. Anderson*	Categorical semantics and category theory.
David Turner *R. Milner, K. Mitchell*	Sorts and polymorphism in the π-calculus.
Stephen Tweedie *E.R.S. Candlin, S. Gilmore*	Efficient operating systems for parallel programs.
Wang Li-Guo *M.P. Fourman, R. Burstall*	Proof-based computation synthesis: its theory, methodology and application; formal methods in design of hardware and software.
Amanda Welsh *R.J. Pooley,* *D.A. Welch (Chemistry)*	Effective software support for chemical research.
Andrew Wilson *K. Mitchell, G.D. Plotkin*	Programming language semantics.

Alexander Mifsud *S.O. Anderson, R. Milner*	Models of concurrency.
Matthew Morley *R. Milner, G. Cleland*	Understanding "safety" in safety critical software and systems through industrially relevant case studies.
Isabel Rojas Mujica *P. Thanisch, R. Pooley*	Performance oriented parallel program design and specification.
Michael Norman *P. Thanisch; D. Wallace (Physics)*	Parallel computing: the applicability of communication costs models in multicomputers; mapping of parallel computations; the relationship between contention in processor interconnects and scheduling models.
Christopher Owens *S. Gilmore, S.O. Anderson*	Assisted synthesis of implementations of functional programs from formal specifications.
Alan Paxton *S.O. Anderson, G.L. Cleland*	High integrity systems design.
Robert Payne *G. Brebner*	Exploitation of fine-grain parallelism using FPGA technology.
Joseph Phillips *R. Candlin, A.S. Wight*	The effectiveness of dynamic process allocation strategies within an occam environment.
Randy Pollack *R. Burstall, M. Fourman*	Computer-assisted proof development, in particular machine-checked mathematics using the Logical Framework and the Calculus of Constructions.
Rob Pooley *E.R.S. Candlin, A.S. Wight*	A unified framework to support cross paradigm performance modelling.
R. Rangaswami *M.I. Cole, K. Mitchell*	Parallelism in functional languages.
Lars Rasmussen *M.R. Jerrum, A.J. Sinclair*	Randomisation techniques in enumeration and optimisation.
Vinod Rebello *D.K. Arvind, R.N. Ibbett*	A performance analysis of self-timed processor design.
Martin Reddy *R.A. McKenzie*	Improving computer graphics in virtual reality systems.
Sandy Robertson *R.N. Ibbett, N.P. Topham*	An object-oriented design environment to simulate and animate microprocessor experiments.

of research projects within the Department unified by their interest in computational theory and its applications. The main areas of interest are logic and proof, concurrency, language and semantics, formal development of programs and systems and computational complexity. The Edinburgh Parallel Computing Centre, which was set up jointly by the Department of Computer Science, the Physics Department and the Edinburgh University Computing Service (EUCS), encompasses the activities of a variety of research groups. It has as one of its main aims the support of applications work by investigating those fundamental aspects of parallel computing which can lead to the development of new and improved systems and user support tools.

The Department is located in the James Clerk Maxwell Building (JCMB), a multi-user building shared by several academic departments including Mathematics, Meteorology, Physics and Statistics together with the main facilities of the EUCS. The JCMB provides extensive accommodation including offices, laboratories, lecture theatres, a library and several common rooms. It is situated in semi-rural surroundings at the King's Buildings campus of Edinburgh University, about 2 miles from the city centre.

Teaching

Undergraduate Courses

The Department offers several Honours and Ordinary degree programmes, as well as providing courses which contribute to degrees in other subjects.

In addition to the Single Honours degree in Computer Science, the following Joint Honours degrees are available: Artificial Intelligence and Computer Science; Computer Science and Electronics; Computer Science and Management Science; Computer Science and Mathematics; Computer Science and Physics; and Computer Science and Statistics. An Honours degree requires four years of study; an Ordinary Degree may be obtained after three years of study. In some cases, applicants with very high qualifications may be exempted from the first year of study.

During the first two years, students study three different subjects, spending only one-third of their time on Computer Science. This is an important element in the education of an Edinburgh graduate, since it provides an opportunity for students to broaden their interests. It also allows students to keep their options open (as far as choice of final degree is concerned) for the first year, and sometimes even the second year, of study. In the final two years, students follow courses in their chosen discipline only. In 1992, a total of 74 students graduated with single or joint Honours degrees.

Degree Philosophy

The basic philosophy underlying the Single Honours Computer Science degree programme is that it should produce graduates of the highest quality. They should have a thorough understanding of the software, hardware, and underlying science, of computer systems, and also have the engineering skills required to design and implement such systems.

Theory and practice are integrated throughout the degree programme. Students undertake a wide variety of practical exercises and projects which reinforce and build on taught material. Communication skills, initiative, professionalism, and the ability to work with others, are also developed as integral parts of the learning process.

The Joint Honours degrees offer students the opportunity to follow part of the Single Honours Computer Science degree programme, in conjunction with another cognate discipline. In the first two years of study, Joint Honours students attend the same Computer Science courses as Single Honours students. Then, in the final two years, the amount of Computer Science material covered is typically half of that in the Single Honours programme, the topics being chosen to form an educationally sound match with the other speciality.

Anna Hondroudakis *R.N. Procter*	Tools for parallel program performance analysis and tuning.
Fred Howell *R.N. Ibbett, N.P. Topham*	Experiments with an object oriented design and simulation environment.
Kayhan Imre *R.N. Ibbett, R. Candlin*	Performance monitoring and analysis of parallel programs.
Ole Jensen *R. Milner*	Theory of higher-order processes and the λ calculus.
Shangjie Jin *G. Brebner, R.N. Ibbett*	Parallel implementation of OSI communication protocols.
Graham Jones *N.P. Topham*	Compilation techniques for high performance architectures.
Roope Kaivola *C.P. Stirling, J.C. Bradfield*	Temporal logics of programs, in particular the modularity aspects of temporal semantics.
Saif Khan *K.N.P. Mitchell, S.O. Anderson*	Machine assisted proofs of operational semantics.
Thomas Kelly *R.N. Ibbett, D.J. Rees*	Towards a unified model of parallel architectures.
Lee Yong Woo *A.S. Wight*	Operating systems, performance modelling and analysis of computer systems.
John Longley *M.P. Fourman, G.D. Plotkin*	Applications of topos theory to programming language semantics.
Christopher Luth *D.T. Sannella, S. Kahrs*	Functional programming and algebraic specifications.
Savitri Maharaj *S.O. Anderson, Z. Luo*	The application of type theory to formal program specification and verification, with a special interest in issues concerning modularity.
Marcus Marr *M.I. Cole*	Algorithmic models of parallel computation.
Paul Martin *R. Candlin, S. Gilmore*	Formal methods in the design of operating systems with hardware monitoring for migration.
Neil MacDonald *S.O. Anderson, M.I. Cole*	Predicting and optimising the performance of multicomputer applications.
Simon McPartlin *R.A. McKenzie, K. Kalorkoti*	Graphical techniques for representing scenes.

David Crooke *R. Candlin, T.H. Heywood*	Methods for architecture-independent parallel software development.
Angus Duggan *R.J. Pooley, K. Mitchell*	The use of structured editing for Literate Programs.
Julyan Elbro *C.P. Stirling*	Semantics and logics of programs.
Richard Eyre-Todd *R.J. Pooley, D.J. Rees*	Tools and techniques for safe data-structure visualisation.
Vashti Galpin *J.C. Bradfield*	Communication and concurrency in CCS.
Marcelo Fiore *G.D. Plotkin, C.B. Jay*	Semantics of programming languages and logics to reason about programs; categorical treatment of non-terminating computations and recursive definitions, and categorical logics for computable functions.
Bo Gao *D.J. Rees, R.A. McKenzie*	Asynchronous VLSI array architectures for algorithm embedding.
Neil Ghani *D.T. Sannella, C.B. Jay*	Applications of type theory and category theory to the systematic development of computer programs.
Ana Goldenberg *R.N. Procter*	Human computer interaction.
Healfdene Goguen *Z. Luo, R.M. Burstall*	Type theory and specification.
Timothy Harris *M.I. Cole, A.J. Sinclair*	Practical and theoretical aspects of the shared memory model for parallel computing.
Rycharde Hawkes *R.A. McKenzie*	Virtual reality.
Roberto Hexsel *N.P. Topham, S.O. Anderson*	Architecture of logically shared, physically distributed memories for scalable multiprocessors; IEEE's Scalable Coherent Interface.
Jane Hillston *R.J. Pooley, J.C. Bradfield*	A compositional approach to performance modelling.
Martin Hofmann *D.T. Sannella, T. Stroup*	Infinite computations and nondeterminism in type theory.

Computer Science is a rapidly evolving discipline, and course and syllabus revision is a continuous activity within the Department. An important influence on the contents of the undergraduate courses is the research being carried out in the Department. In the final two years, each specialist course is normally taught by a member of staff who is involved in research in the area. In the final year, each student undertakes a major individual project, and the topics covered are usually closely linked to research work in the Department. Also, final year students in particular are encouraged to attend the research seminars given by both internal and external speakers throughout the year.

Computer Science Courses

The first and second year courses in Computer Science provide a thorough grounding in the structure and organisation of present-day computing systems, and in the production of quality computer software using modern development environments. The first year course is organised so that students with adequate previous programming experience can study various commercial and industrial computer applications instead of attending introductory programming classes.

The third year of the Single Honours Computer Science programme builds on the knowledge and skills acquired in the first two years, and involves studying core Computer Science areas in depth. Students study eight of the taught specialist topics and a Professional Issues module, and also undertake two large practical projects: one individual project and one group project. Third year courses are currently offered in the following topics:

> Algorithms and Data Structures; Computability and Intractability; Computer Architecture; Computer Communications; Computer Design; Database Systems; Knowledge Based Systems; Language Semantics and Implementation; Operating Systems; Professional Issues; and Programming Methodology.

The fourth year involves advanced study of both familiar and new areas. Students spend about half of their time on a major individual project, and the remainder on studying six of the taught specialist topics. Fourth year courses are currently offered in the following topics:

> Communication and Concurrency; Compiling Techniques; Computational Complexity; Computer-Aided Design; Computer Algebra; Computer Graphics; Denotational Semantics; Distributed Systems; Human Computer Interaction; Microprocessor Design (practical course); Parallel Architectures; Program Logics; Computer Simulation and Modelling Techniques; Software for Parallel Computers; and VLSI Design.

The final degree classification is determined on the basis of performance in examinations, practical work, and project work over the two final years of study; with the student spending approximately half their final year on an individual project which counts as 40% towards that years assessment.

Accreditation

The Department is visited at intervals of (normally) five years by an Accreditation Panel of the British Computer Society. Graduates from courses accredited by the Society may be exempted from Parts I and/or II of the Society's Professional Examinations, and may be recommended by the BCS as appropriate for Chartered Engineer status (C.Eng.). Candidates for professional Membership of the Society and Chartered Engineer status are required to have passed or been exempted from Parts I and II and subsequently to have completed six years of monitored work experience.

The current levels of accreditation are:

B.Sc/B.Eng. (Hons) in Computer Science	Parts I & II and C.Eng.
B.Sc. (Ord) in Computer Science	Part I[a]
B.Sc. (Hons) in Artificial Intelligence and Computer Science	Parts I & II and C.Eng.
B.Eng. (Hons) in Computer Science and Electronics	Parts I & II and C.Eng.
B.Sc. (Hons) in Computer Science and Management Science	Part I
B.Sc. (Hons) in Computer Science and Mathematics	Part I
B.Sc. (Hons) in Computer Science and Physics	Part I

[a] Subject to passing CS3 Programming Methodology and 1 Major Practical

The B.Eng in Computer Science and Electronics is also accredited by the IEE for student intakes up to and including 1992. Graduates from this course with first or second class honours degrees meet the educational requirements for corporate Membership of the IEE.

MSc in Computer Science

The Department offers an MSc in Computer Science with three themes: Computer Systems Engineering, Parallel Systems, and Theoretical Computer Science. The course is accredited by the British Computer Society and carries exemption from the BCS Part II Examinations.

The course runs from October to September. During the first half of the academic year, eight chosen lecture modules are studied and examined, four of which must be 'core' modules within one of the themes. The following six months are spent on a full-time project, examined by dissertation, usually supervised within the department[1]. Closing date for MSc applications is 31 March 1994.

[1] Further MSc information may be obtained from Ms M Davis at the Departmental address, or tel. 031 650 5175/5129, fax. 031 667 7209, e-mail: msc_enq@uk.ac.ed.dcs

Postgraduate Students

Student/Supervisor	Project
Shaikha Al-Jabir *R. Candlin, R. Pooley*	Distributed systems and parallel processing systems
Thorsten Altenkirch *R.M. Burstall, Z. Luo*	Type theoretic program verification; inductive types.
David Aspinall *D.T. Sannella, B.C. Pierce*	Foundations for computer-assisted proof-checkers.
Geoffrey Ballinger *S.O.Anderson;* *J. Ponton, René Bañares-Alcántara (Chem. Eng.)*	The semantics of modelling languages.
Andrew Barber *G.D. Plotkin*	Classical linear logic and concurrency.
Maria Cristina Boeres *P. Thanisch, G. Brebner*	Distributed simulation; parallel processing; distributed algorithms.
Glenn Bruns *C.P. Stirling, G.L. Cleland*	Abstraction techniques for the verification of temporal properties of processes.
Albert Burger *P. Thanisch, S. Gilmore*	Concurrency control and recovery mechanisms in distributed and parallel database systems; object-oriented database systems.
Louise Burnham *M.P. Fourman*	Formal specification and verification of systems.
Claudio Calvelli *K. Mitchell, R. Milner*	Relationships between π-calculus and λ-calculus.
Pietro Cenciarelli *R.M. Burstall, G.D. Plotkin*	Categorical semantics and program logics.
Conrad Chin *R. Candlin, M.P. Fourman*	Distributed ML and the semantics of programming languages.
Søren Christensen *C.P. Stirling, F. Moller*	Models for distributed systems; formal frameworks for describing concurrent and distributed systems.
Alan Crawford *N.P. Topham, R.N. Ibbett*	Design of high performance decoupled architectures; compilation techniques for decoupled architectures.

Visiting Scholars 1992-3

Dr P. Bird: *(ACRI, Lyons)*
Computer architecture; compilation.

Dr Z. Esik: *(Jozsef Attilla University, Szeged)*
Iteration theories and concurrency.

Dr Y. Hirshfeld: *(University of Tel-Aviv)*
Mathematical logic and non-standard analysis; concurrency; Computer Science logic.

Dr H. Hüttel: *(University of Aalborg, Denmark)*
Concurrency theory.

Mr Z. Jianping: *(Southwest Institute of Atomic Research, Sichuan, PR China)*
Computer communications; distributed systems.

Dr Y. Kinoshita: *(Electrotechnical Laboratory, Ibaraki, Japan)*
Computer-aided verification; type theory; category theory.

Mr N. Mylonakis: *(University of Barcelona)*
Algebraic Spefification.

Dr S. Pelagatti: *(Universita di Pisa)*
Parallel programming and implementation; performance modelling.

Dr F. Seredynski: *(Polish Academy of Sciences, Warsaw)*
Process allocation for parallel machines.

Mrs Chen Shuyun: *(The Institute of Aeronautic Computation Technology, PR China)*
Computer architecture; simulation techniques.

Dr D. Spooner (dxs): *(University of Calgary)*
Categorical methods in concurrency.

Prof A. Tarlecki: *(Polish Academy of Sciences, Warsaw)*
Formal specification and development of programs.

Prof R.D. Tennant: *(Queen's University, Ontario)*
Semantics and design of programming languages and logics.

Ms V. Vuleva; *(University of Rousse, Bulgaria)*
EC/*TEMPUS* Visitor
HCI; "intelligent" systems.

Dr D. Wu: *(Nanching University of Jiangxi)*
Parallel systems.

Most lecture modules have practical work associated with them. The modules currently offered are:

Computer Systems Engineering

ASIC Design; Behavioural Specification and Verification; CAD/CAM; Systems Integration; Systolic Design

Parallel Systems

Concurrent Architectures; Principles of Parallel Computing; Distributed Systems; Software for Parallel Computers

Theoretical Computer Science

Algebraic Complexity; Applicative Programming and Specification; Communication and Concurrency; Computational Complexity; Computer Aided Formal Reasoning; Formal Programming Language Semantics

General

Software Systems Concepts; Graphics; Operating Systems; Database Systems

Industrial Contacts

The department has rich contacts with local and national industry. At the level of teaching we involve industrial concerns in mapping the direction of our teaching. The Information Technology Education Advisory Board meets annually to review the past year and comment and advise on new proposals for teaching innovations. As a response to this involvement a number of companies sponsor prizes for good performance in various components of the course.

At a different level, the department mounts a number of industrial courses each year. The aim of these is to transfer technology and techniques developed within the department to industry as rapidly as possible.

Prizes, Bursaries and Studentships

Computer Science 4

Sun Microsystems : Prize for Best Single Honours student.

Ford Motor Company : Prize for Best Artificial Intelligence & Computer Science student;

Hewlett Packard : Prize for Best Computer Science & Electronics Student.

Computer Science 3

FI Group : Prize for Best Single Honours student;

Hewlett Packard : Prize for Computer Science & Electronics Student;

Anderson Consulting : Prize for Best Group Project;

Thorn EMI (CRL) : Bursary.

The CS3 Professional Issues lecture series has been supported by **Andersen Consulting, An Teallach, Logica and Salomon Brothers**

Computer Science 2

Proctor & Gamble : Prize for Best Student.

Computer Science 1

Prentice Hall Publishers : Prize for Best Student.

The following companies are represented on the IT Education Advisory Board. In choosing the membership of the board we attempt to balance small concerns with larger companies and locally based industry with a range of national companies. In this way we hope we can balance the course teaching to meet the priorities of a wide range of the IT industry and beyond.

Digital (Scotland), Ford Motor Company, GEC Ferranti
Hewlett Packard, Hitachi Europe, IBM, ICL, Intelligent Applications
Logica, Sharp Laboratories, Spider Systems, Syntek
Wolfson Microelectronics.

Ms J.J. Smith (jenny): Computing Support Officer
faults; support; accreditation; backups.

Ms R. Soutar (rs): p/t Computing Officer
administrative databases; HP support.

Mr R.W. Thonnes (rwt): C compiler; hardware/software systems development; Sparse vector machine.

Mr J.S. Turnbull (jst): IS1 co-organiser (with AS Wight); Mac; Apple fileservers and gateways; animator generator; c-prolog; ML teaching support.

Administrative and Secretarial Staff

Mrs T.L. Combe (tlc)	LFCS Secretary
Miss A. Noble (ano)	Enquiries and General Office
Ms L.M. Edgar (lme)	Secretary to Professor Fourman
Mr S.S. Falconer (ssf)	EPCC/CS Accounts
Ms M.E. Curran (mec)	Secretary to Head of Department
Miss E.A. Kerse (eak)	Secretary to Professors Burstall & Plotkin
Ms M. Lekuse (mkl)	LFCS Administration
Ms M. Davis (mda)	Secretary to Professor Milner
Ms L.M. Paterson (lmp)	CS Administration
Mrs L.M. McGill (lmm)	Temporary CS Administration (September 1993-August 1994)

Technical Staff

Mr J.C. Dow (jcd)	Laboratory Superintendant
Mr A.M. Duncan (amd)	General workshop
Mr M.L. Graham (mlg)	APM Production
Mr D.C. Hamilton (dch)	Workshop Chief Technician
Mr G. Inkster (gi)	General workshop
Mr J. Johnstone (jj)	Hardware systems development
Mr P.J. Lindsay (pjl)	Special systems
Mr T.S. Wigham (tsw)	General workshop

Mr R.A. Pollack (rap): Machine-checked mathematics; type theory; proof theory; design and implementation of mechanized proof systems.

Dr J. Power (ajp): Category theory and logical frameworks.

Dr D. J. Pym (dpym): Logical and semantical frameworks.

Dr C. Raffalli (cr): Type for proof and programs.

Dr D. Sangiorgi (sad): Semantics of concurrent systems.

Dr G. Sorkin (gxs): Randomised Algorithms.

Dr R.T. Stroup (rts): Foundations of formal program development and verification; foundations of concurrency theory.

Computing Officers

Mr P. Anderson (paul): LFCS Systems Development Manager.

Mr D.W.T. Baines (dwb): CS2/3/4/MSc systems support; CS2 teaching support; news; mail; communications support; Gandalf.

Ms J.T. Blishen (jtb): Unix system management; sources server; information distribution; CS1 teaching environment.

Mr J.H. Butler (jhb): Service Manager; EPCC liason; PC-NFS; Maple.

Dr G.L. Cleland (glc): Assistant Director of LFCS.

Mr C. Cooke (cc): Staff systems manager; general system development; Emacs editor support.

Ms C. Dow (carol): p/t Computing Support Officer
Faults; documentation; advisory; accreditation.

Ms M. Findlay (morna): LFCS System Manager; mail.

Mr A. Howitt (arch): Hardware laboratory support; CS3/4 projects; P-CAD; IBM PC clones.

Mr D.J. Rogers (ddr): VLSI support; SOLO; Cadence; ELLA; Faculty representative for ECAD, Eurochip and EASE.

Dr G.D.M. Ross (gdmr): CS1 teaching cluster; Sun; X11; networking; mail; X toolkits.

Mr A.J. Scobie (ajs): MaPS Project (Faculty); APM replacement; annexes; Unix device drivers; technical support to Service Manager, EUCS liaison.

Research

Computer Science is a subject which is less easily compartmentalised than most other scientific and engineering disciplines, and so although researchers in the Department form themselves into groups such as the Complexity Group, the Computer Systems Group, the Performance Group, and the various 'clubs' within the Laboratory for Foundations of Computer Science (which itself organises a coherent subset of the research activities of the Department as a whole), there is considerable cross representation, and the research of any one group may well draw on the experience and expertise of others. Research in computer systems, for example, involves a number of interrelated activities, including work on architectures, networks, languages, etc., and the use of VLSI, no one of which can be carried out independently of all the others. The division of research into the areas described here does not therefore represent a strict compartmentalisation of activities within the Department. It shows, rather, the major emphases of one or more projects contributing to the overall programme of computer science research. After each section there follows a short list of members of staff involved in the area. This indicates some of the principal workers in ther area, it is not intended to be exhaustive.

Applications

Computational Nanotechnology

Molecular nanotechnology is predicted to remake technology from the bottom up via the explicit control of atoms and molecules as building blocks. Nanoscale molecular machines will be able to guide the placement of molecules and atoms, enabling the construction of atomically precise materials and devices. While there is debate about the time frame for developing molecular manufacturing capabilities, it is clear that computational tools can substantially reduce the development time. Molecular computer aided design and modeling (CAD/CAM) software, and related specification tools, will allow "molecular engineering" to be planned and analysed via computer before construction is undertaken, just as in other engineering fields. Research in this area involves the development of computational chemistry tools for specifying, designing and modelling molecular machines at the nanoscale level.

Todd Heywood

Human Factors in Computer Systems Design

Human factors addresses the problems inherent in matching the functionality of computer systems with the needs of their users. Understanding users and their working environment is critical to the success of any computer system.

Human factors research has been instrumental in the improvements in system usability achieved over the past decade. There are still significant gaps, however, in our understanding of user interface design requirements for many specialised application environments. Moreover, advances in the *individual* user interface, and the proliferation of desktop computing, have served to highlight the importance of the *group* user interface.

On a larger scale, the problems of ensuring that IT fully addresses business objectives are growing in complexity. A recent survey revealed that only a small proportion of commercial IT investment currently leads to successful applications.

Human factors research is by nature inter-disciplinary. It covers themes ranging from technologies and techniques for interaction, through user performance and behaviour, to design methodologies, the study of innovation, and industrial sociology.

Currently active areas of research within the department include:

- The design of user interfaces and systems for computer-supported collaborative work. Recent research has been studying the impact of computer mediation on group processes such as the sharing of information and coordination of activities.
- User interfaces for medical imaging applications. Work has just commenced on a collaborative project to design of a user interface for a system to assist radiologists in the viewing and interpretation of mammograms.
- Improved user interfaces for parallel program development and performance analysis tools. This work is a collaboration with the Edinburgh Parallel Computing Centre.
- Organisational factors in the design and implementation of IT systems. This research is a collaboration with the University's Research Centre for Social Sciences. A major recent project focused on the problems companies face in meeting the growing demand for specialist IT expertise, and integrating it effectively with the more traditional forms of business expertise such as that held by managerial and administrative staff.

Rob Proctor

Complexity and Algorithms

Computational Complexity is the quantitative study of the 'inherent difficulty' of solving computational problems. The study is motivated by the empirical observation that the resources required to accomplish various computational tasks vary dramatically between tasks. The meaning of 'resource' depends on the setting, but typical examples are time, space or hardware size. The study is wide-ranging, and its techniques rest on increasingly sophisticated mathematics.

(Senior) Research Workers/Postdoctoral Fellows 1992-93

Mr G. Bruns (bruns): Temporal logic, concurrency theory and their application to the design of safety-critical systems.

Dr J Esparza (je): Concurrency; model checking; petri nets.

Dr P.A. Gardner (pag): Formal specification of logics.

Dr K. Goossens (kgg): Theorem provers in general, their user interfaces, encoding semantics of (hardware description languages), tableaux methods.

Mr T. Harris (tjh): Practical and theoretical aspects of the shared memory model for parallel computing.

Dr C. Jones (ccmj): Formalising mathematics, semantics, tools for inductive types.

Mr G. Jones (gxj): Compilation and optimisations techniques for high preformance architectures.

Dr S Kahrs (smk): Functional programming, term rewriting, semantics of program specification.

Dr T. Le Sergent (tls): Implementation of concurrent functional programming.

Dr Z. Luo (zl): *p/t lecturer*
Type theories and logics; general proof theory; model theory and semantics; program specification and development; computer-assisted reasoning.

Mr B. McConnell (bm): Formal integration of concurrent systems.

Dr A.J. McIsaac (ajmi): Foundations of formal system design.

Dr J.H. McKinna (jhm): Category theory; type theory.

Mr R. Marlet (renm): Semantics of programming languages; Partial evaluation

Dr D. Matthews (dcjm): Standard ML; Poly/ML; compilers for functional languages; concurrency, particularly concurrent implementations of Standard ML.

Dr F. Moller (fm): Process calculi; process decomposition and its application to the decidability and verification of concurrent systems; timed models of concurrency; the Concurrency Workbench.

Dr B.C. Pierce (bcp): Typed λ-calculi; Type-theoretic foundations for object-oriented programming languages; programming languages based upon Milner's π-calculus.

Dr R.N. Procter (rnp): *(Director of Studies)*
Human-Computer Interaction, especially interfaces for medical imaging applications and group-based work. Organisational factors in systems design and design methodologies; social shaping of Information Technology.

Dr A.J. Sinclair (ali): [7] Complexity theory, design and analysis of algorithms, randomised computation and probabilistic algorithms, approximation algorithms for combinatorial problems.

Dr A.D. Smaill (smaill; smaill@uk.ac.ed.ai.sb): *(CS/AI Tutor)*
Theorem proving, automated inference and its control; constructive logics.

Mr F. Stacey (fs): *(Director of Studies; Undergraduate Admissions Officer)*
All aspects of distributed operating systems, particularly the provision of file services; the specification and verification of (parallel) algorithms used in the implementation of such services.

Dr P. Thanisch (pt): Parallel computing systems — load balancing; image analysis; database systems.

Dr N.P. Topham (npt): *(CS3 Course Organiser)*
Computer architecture, and parallel computers; the design of high-performance computing systems, architecture simulation tools, and quantitative analysis of high performance systems.

Dr A.S. Wight (asw): *(IS1 Course Organiser; Overseas Student Exchange Programme Co-ordinator)*
computer communications networks; workload characterisation.

Honorary Fellows

Mr P.D. Schofield (pds): The use of computers in teaching, assessment and departmental administration particularly the automatic marking of and detection of plagiarism in practical assignments; data structures — sorting and searching algorithms.

Prof. Dr. Christian Lengauer (lengauer@de.uni-passau.fmi): Infusion of Parallelism into specifications and programs, parallelizing comilation, regular and systolic array design, formal semantics of parallelism. Program transformation and refinement, programming language design and semantics. Applications of automated theorem proving. (Residing at the University of Passau.)

[7] Until January 1994; thereafter at University of California, Berkeley

Algebraic Complexity

Algebraic complexity studies intrinsic requirements in the computation of algebraic functions, such as matrix multiplication, polynomial evaluation and interpolation. A machine independent model is used in which the basic operations are addition, subtraction, multiplication and division (sometimes extended or even contracted). Most work assumes a uniprocessor, although parallelism is also studied.

The most highly developed area is concerned with bilinear forms. However, despite substantial progress, non-linear lower bounds are still elusive. For example, the best known lower bound for matrix multiplication over arbitrary fields is linear and has seen no improvement for at least ten years (all the advances in this much studied topic have been on upper bounds).

The only general technique for non-linear lower bounds has its basis in Algebraic Geometry. This has yielded optimal results for such problems as polynomial interpolation, but in the case of bilinear forms it is of no help. There is a pressing need for more techniques, especially ones which can handle bilinear problems.

Kyriakos Kalorkoti

Algorithms and Data Structures

In contrast to algebraic complexity, the problems here are combinatorial, and the model of computation is machine-based. The problems considered come from many areas, including graph theory, combinatorial enumeration and geometry. A particular concern at Edinburgh has been the study of *randomised* algorithms whose progress is influenced by the outcome of a sequence of coin tosses. Is it possible that randomised algorithms have greater computational power than deterministic ones? Surprisingly the answer appears to be 'yes', and several probabilistic algorithms have been discovered which are faster than the best known deterministic counterparts.

Recent work has focused on randomised algorithms which involve the simulation of an appropriate Markov chain. Such algorithms arise in the analysis of systems in statistical mechanics, and in stochastic optimisation techniques such as simulated annealing. Moreover they represent the only known class of efficient approximation algorithms for a number of problems in combinatorial enumeration.

Mark Jerrum, Alistair Sinclair, Sigal Ar

Computer Architecture and Computer Systems

The Department has been active in computer systems design for many years and has expertise in operating systems, language support and hardware design and implementation, particularly in relation to local area networks, and personal workstations. At any one time a variety of small projects are undertaken in this general

area, together with one or two more major ones. Interest is focusing increasingly on parallel systems and components for constructing them.

The Sparse Project

The Sparse Project is concerned with the design and implementation of hardware and software for a sparse vector processing system. The hardware is incorporated into an existing workstation where it acts as an accelerator providing specialist support for the processing of sparse vectors. The project also involves work on compilers and on language extensions to give programmers access to the sparse vector support mechanisms provided in hardware. The applicability of these mechanisms to other types of computing is also being investigated.

Tim Hopkins, Roland Ibbett, Nigel Topham

Architectural Simulation

A computer architecture and its implementation can be represented in a number of ways and at different levels of abstraction. At the highest level are multiprocessor systems, while at the lowest an architecture consists of an ensemble of transistors on a VLSI chip (or set of chips). A variety of simulators has been designed for each different level of abstraction, and some design tools can generate implementations automatically at lower levels. The Hierarchical Architectural design and Simulation Environment (HASE) project aims to allow designers to create architectural designs at different levels of abstraction, to explore designs through a graphical interface system and to invoke the appropriate simulator at each level. Working at the processor architecture level, for example, a designer could investigate bottlenecks in the flow of instructions and try alternative design strategies to eliminate them.

The component parts of a computer can be treated very naturally as objects, and so the environment is being created using the object oriented simulation language Sim++. The environment will include object libraries, design editors etc., and instrumentation facilities to allow performance measurements to be derived from simulation runs.

Roland Ibbett, Nigel Topham, Todd Heywood, Murray Cole, Pat Heywood

High-Performance Computer Architecture

Computer architecture is a continually evolving discipline. It embraces a wide spectrum of activities in computer science: from the design of hardware components, and the technology of integrated circuits, through the organisation and structure of computing systems, to compiler and language issues, and ultimately to the applications which use the resulting computer systems.

dynamic behaviour of parallel programs; the effectiveness of process migration and allocation strategies, and the design of operating systems for their support.

Dr M.I. Cole (mic): *(CS1 Course Organiser)*
The design and implementation of programming languages and frameworks for parallel computers; the use of the Bird-Meertens theory of lists as a foundation for parallel program design and transformation.

Dr S.D. Gilmore (stg): *(CS4 Course Organiser; CS/Mathematics & CS/Statistics Tutor)*
Formal methods of program development; formal specifications; software engineering; software tools; specification of concurrent systems.

Dr T.H. Heywood (thh): *(CS4 Projects)*
Parallel computing systems, particularly topics bridging the gap between theory and practice; models of parallel computation; decentralized control of widely-distributed systems; computational nanotechnology.

Dr T.M. Hopkins (tmh): *(CS2 Course Organiser; CS/EE & CS/Physics Tutor)*
Special-purpose and high-performance processor design; sparse matrix and vector processing.

Dr M.R. Jerrum (mrj): *(Nuffield Fellow for one year from October 1993)*
Computational complexity — data structures, efficient algorithms (including those involving randomisation), resource bounded complexity classes; combinatorial mathematics — random graphs; quantitative treatment of rates of convergence in stochastic systems; learning theory.

Dr K. Kalorkoti (kk): *(Director of Studies)*
Computational complexity with special interest in algebraic complexity; computer algebra; decision problems in group theory.

Dr Z Luo (zl): *(part-time lecturer)*
Type theories and logics; general proof theory; model theory and semantics; program specification and development; computer-assisted reasoning

Mr R.A. McKenzie (ram): *(Director of Studies)*
Computer graphics — algorithms for scene rendering and parallelisation, response times in virtual reality systems; Computer architecture — specialised processor systems for graphics and imaging; VLSI design

Dr K.N.P. Mitchell (kevin): *(Director of Studies)*
Functional programming language design and implementation - the application of operational semantics to compiler generators.

with fixed points and their applications to verification and description of program properties.

Dr D.T. Sannella (dts): *(SERC Advanced Research Fellow)*
Algebraic specification and formal program development, mechanised reasoning, programming methodology and functional programming languages.

Senior Lecturers

Mr S.O. Anderson (soa): *(Chairman Teaching Committee, Organiser, Postgraduate Lectures)*
Use of programming logics in the specification and development of verified programs; the use of Martin-Lof's type theory and the provision of proof assistants for the theory; application of formal methods in safety-critical systems.

Mr R.J. Pooley (rjp): *(CS/Man. Sci. Joint Degree Tutor)*
Simulation methodologies; support environments for modelling and design; graphical programming and visualisation for modelling and design; performance modelling; integration of quantitative and qualitative system design approaches; object-oriented language design and implementation.

Dr D.J. Rees (djr): *(MSc Course Organiser)*
Computer Aided Design tools for VLSI design; silicon compilation from behavioural descriptions; hardware description languages; silicon assembly; novel VLSI architectures; configurable cellular arrays.

Lecturers

Mr D.K. Arvind (dka): *(MSc Projects Co-ordinator)*
Formal integration of concurrent systems; algorithms and environments for parallel computation; parallel and distributed simulation.

Dr J.C. Bradfield (jcb): *(MSc Admissions)*
Modal logic and model checking for infinite systems; computer-assisted verification techniques.

Dr G.J. Brebner (gordon): *(Deputy Head of Department; Director of Studies)*
Computer networking, protocol specification and verification, distributed computing environments; configurable cellular arrays; computational complexity.

Dr E.R.S. Candlin (rc): *(MSc "Director of Studies")*
Cost models for different classes of parallel program; the statistics of the

Research in the area of high-performance computing within the Computer Science department at Edinburgh is currently centred around the notion of decoupled architectures. This is an implementation technique for high-performance systems in which the activities of control-flow management, address generation and operation evaluation are partitioned across closely coupled and highly specialised sub-processors. These represent a form of MISD architecture, and their principal goal is to overcome the ever-increasing discrepancy between processor cycle times and memory latency. Work in this area involves looking at not only the structures out of which one can construct such architectures, but also at the decoupling behaviour of applications and the ways in which new compilation strategies can be used to optimise the run-time characteristics of decoupled systems.

Tools for investigating decoupled systems are under construction, and include both a prototype compilation system and architecture simulators. These are being developed in order to discover the extent to which real-life applications can be decoupled, and the extent to which compilation decisions for decoupled architectures can be deferred until run-time. Effectively, the memory system in a decoupled architecture behaves as if it were part of the processor's pipeline, and with large memory latencies there may be data dependencies within this extended pipeline. Such dependencies interfere with the pipeline flow, and therefore need to be avoided. However, in many cases, these dependencies cannot be resolved at compile time but could be resolved straightforwardly at run-time. This is a problem similar to, but subtlely different from, the traditional vectorization decision problem, and new approaches to finding efficient solutions are being sought.

Nigel Topham, Tim Harris, Graham Jones

Computer Graphics

Current research interests include the application of Binary Space Partition trees, computational geometry, virtual reality and bidirectional ray tracing. Work on BSP trees concentrates on complexity issues of constructing good trees and in restructuring trees for dynamic scenes. Computational geometry work also addresses handling of animated scenes. In both of these cases we investigate strategies for parallelisation on different architectures. In virtual reality systems we investigate the very real problem of response times. The aim is to construct real-time optic and haptic environments for experiments in human perception and control in collaboration with the Department of Psychology here at Edinburgh. The investigation will include examining which features of the optic array are necessary for particular applications and will look at the fundamental principles for the efficacy of virtual reality systems. The Department is a founder member of the Edinburgh Parallel Computing Centre (EPCC) which provides a variety of very powerful architectures. Visualisation is a major consumer of graphics work within the EPCC. There are currently two research projects following different approaches to bidirectional ray tracing being conducted jointly with EPCC.

Eric MacKenzie

Concurrency: Theory and Tools

Edinburgh's work on models for concurrent communicating programs and hardware arose in the 1970s, in the course of trying to develop an all-embracing semantic theory for computation. It was soon found that existing general theories did not naturally embrace concurrency. This led to the theory of power domains, and to the Calculus of Communicating Systems (CCS). In the past few years this foundational work has been broadened to include new elements, both theoretical and practical. On the theoretical side, significant progress has been made with various logical approaches and the algebraic theory has been enriched to include mobile processes. On the practical side it has led to the development of the Concurrency Workbench and there has recently been increased use of process calculi in the analysis and design of industrial systems.

π-Calculus and Action Structures

The π-calculus is a prototypical calculus for the description of *mobile systems*, i.e. systems with a dynamically changing communication topology. Studies on π-calculus may be classified into two main streams: on the one hand, the extension to π-calculus of the theory developed for CCS, the process algebra on which π-calculus is founded ; on the other hand the investigation of aspects more specific to mobility.

The π-calculus aims to be canonical for computations with processes, much in the same way as λ-calculus for computations with functions. A consistent effort has been dedicated to the comparison between the two. This includes the study of encodings of λ-calculus evaluation strategies and dialects into π-calculus, and the analysis of the equivalence on λ-terms induced by such encodings.

More recently, the study of synchronous versions of π-calculus, in which a transition may involve arbitrarily many "particulate" interactions, has been the trigger for work on *action structures*. These are proposed as a mathematical structure which can underline concrete models of computation, but which is free of the ad hoc details which are often present in such models; if successful they may play a role analogous to Scott's domains for sequential computations, though they have a more intensional nature in that they may (but need not) present dynamics explicitly.

Robin Milner, Davide Sangiorgi, Benjamin Pierce

Concurrent ML

In recent years, the programming language Standard ML has gained in popularity. The publication of the formal definition of Standard ML and the availability of

People

The department is a comprises a large community of staff, postgraduate and undergraduate students with diverse interests. The following lists cover the staff and postgraduate students.

Members of Staff

Professors

Professor R.M. Burstall (rb@dcs.ed.ac.uk): [6] *(Director, LFCS)*
Computer-aided proof and its relation to type theory; development of correct programs with respect to a specification; applications of category theory in computer science.

Professor M.P. Fourman (mikef): *(Co-Director, LFCS; Chairman Syllabus Committee)*
System specification, verification and synthesis — formal models of digital behaviour and of the design process; temporal and data abstraction as used in digital design; design and implementation of formally-based system-design tools; proof and proof assistants; specification languages and their semantics.

Professor R.N. Ibbett (rni): *(Head of Department; Associate Director, EPCC)*
Computer and network architectures; design, simulation and performance evaluation of parallel and novel computer systems.

Professor A.J.R.G. Milner (rm): *(FRS; FRSE; Turing Award Winner, 1991; SERC Senior Research Fellow; Co-Director, LFCS)*
Mathematical models of computation — models and calculi for concurrent computation; how to present logics to machines; semantics of high-level programming constructs.

Professor G.D. Plotkin (gdp): *(FRS; FRSE; SERC Senior Research Fellow; Co-Director, LFCS)*
Applications of logic, especially denotational and operational semantics of programming languages; semantics of type systems, monadic theories of computation, general proof theory and the semantics of natural language, particularly type-free intensional logics for situation theory.

Readers

Dr C.P. Stirling (cps): *(Co-ordinator — Phd Admissions and Thesis Proposal Reviews)*
Models and calculi for concurrent computation; modal and temporal logics

[6] Subsequent entries provide usernames only

James Hugh McKinna
Price – £9.00

CST-97-93 A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification
Michael Mendler
Price – £8.50

CST-98-93 PAC-Learing Geometrical Figures
Paul W Goldberg
Price – £8.50

CST-99-93 Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms
Davide Sangiorgi
Price – £8.00

CST-100-93 Embedding Hardware Description Languages in Proof Systems
Kees G W Goossens
Price – £9.00

CST-101-93 Timed Processes: Models, Axioms and Decidability
Liang Chen
Price – £8.00

CST-102-93 Task Assignment in Parallel Processor Systems
Sathiamoorthy Manoharan
Price – £8.00

high-quality compilers on many diverse platforms has helped promote interest in ML. Recently, there has been a lot of interest in concurrent languages and machine architectures, so a natural question to ask is how we can integrate concurrency primitives with ML. A number of different researchers have tackled this problem. Some have been inspired by CCS (such as the language PFL). Others are based on communication through shared memory.

An initial implementation of concurrent ML operated on a single processor using task switching to simulate concurrency. A true concurrent implementation was developed for the DEC Firefly machine (a shared memory multi-processor machine). A much greater challenge is the implementation of concurrent ML on a distributed system. This is currently being undertaken for a network of Sun workstations. An implementation for the EPCC Meiko computing surface is being planned.

Robin Milner, Mike Fourman, David Matthews, Theirry Le Sergent

Temporal Logics

Work on temporal logics comprises a mix of theoretical and more applicable investigations, concentrating largely on the modal mu-calculus. Theoretical work has included a number of complete axiomatizations of modal and temporal logics. Questions of expressiveness and definability have also been tackled. On the more applicable side, a strong theme for the last few years has been the development of model-checking techniques—that is, determining which states in a transition system satisfy a formula of some temporal logic. Tableau-based techniques were developed for model-checking finite and infinite-state systems in the propositional modal mu-calculus. An important aspect of this is its locality (using, so far as possible, only information local to a state in the checking of that state). Applications to CCS, Petri nets, and concurrent **while**-languages have been investigated, case-studies performed, and a tool developed.

Colin Stirling, Julian Bradfield

Decomposition on Infinite State

The method of decomposition is an example of the art of divide and conquer. It may support verification techniques it may also support decidability and axiomatizability of process equivalences on systems with potentially infinitely many states.

In the setting of process algebra, examples of infinite-state systems include BPA (Basic Process Algebra), and BPP (Basic Parallel Processes). In both these cases bisimulation equivalence is decidable. Currently, work is taking place on finding ways of decomposing other (stronger) models of computation.

Equational Theories

Given a particular calculus for expressing processes, and a particular semantic congruence relation between process terms, an important goal in the development of the theory is the search for a sound and complete equational axiomatisation for the congruence. The success of this endeavour provides two important aspects: firstly, equational theories provide an elegant verification technique; and secondly, the axioms of the syntactic theory can provide enlightenment as to the properties of the semantic relation.

Colin Stirling, Julian Bradfield, Faron Moller, Javier Esparza

Software tools

The main tools for the verification of concurrent systems in are the Concurrency Workbench and a Proof Assistant for infinite-state systems. A translator to allow the verification of value-passing CCS programs with the Workbench is also available.

The Concurrency Workbench, a set of formally based tools for analysing and designing concurrent systems, has been built by a collaboration between Edinburgh and Sussex Universities and the Swedish Institute of Computer Science. In addition to a number of facilities for examining the operational behaviour of CCS terms, the Workbench incorporates equivalence checkers and preorder checkers that test for various relations between CCS terms. Also it includes a temporal logic model checker that tests for temporal properties of CCS processes. Furthermore, the system includes a feature which supports the hierarchical development of complex systems by means of interactive equation solving. Recent extensions to the Workbench support timed behaviour (with Temporal CCS and extensions to the mu-calculus), the synchronised calculus SCCS, and the output of process definitions in various formats used by other verification tools.

For infinite systems, a computer-assisted proof tool has been developed that is based on a tableau proof system. The user must provide well-foundedness conditions required to prove liveness properties. However, the tool applies those tableau rules that require no user intervention, and checks the correctness of user-applied rules. The modular design of the tool allows different models of systems; the current implementation supports petri nets.

Julian Bradfield, Faron Moller, Glenn Bruns, Colin Stirling

Formal Development of Programs and Systems

Specifying and Developing ML Programs

Extended ML is an algebraic specification language for specifying Standard ML programs. Specifications in Extended ML look just like programs in Standard ML

ECS-LFCS-93-265
Errata and Remarks
David J Pym

ECS-LFCS-93-266 (also published as CST-99-93)
Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms
Davide Sangiorgi
PhD Thesis – price £8.00

ECS-LFCS-93-267
Correctness of Data Representations in Algol-like Languages
R D Tennent

ECS-LFCS-93-268 (also published as CST-100-93)
Embedding Hardware Description Languages in Proof Systems
Kees G W Goossens
PhD Thesis – price £9.00

ECS-LFCS-93-269
The Formalisation of a Hardware Description Language in a Proof System: Motivation and Applications
Kees G W Goossens

ECS-LFCS-93-270
A Theory of Bisimulation for the π-calculus
Davide Sangiorgi

ECS-LFCS-93-271 (also published as CST-101-93)
Timed Processes: Models, Axioms and Decidability
Liang Chen
PhD Thesis – Price £8.00

ECS-LFCS-93-272
Uniform sampling modulo a group of symmetries using Markov chain simulation
Mark Jerrum

ECS-LFCS-93-273
Structure and Behaviour in Hardware Verification
K G W Goossens

Theses

CST-95-92	Correctness Proofs of Compilers and Debuggers: an Approach Based on Structural Operational Semantics Fabio Q B da Silva Price – £9.00
CST-96-92	Deliverables: a categorical approach to program development in type theory

ECS-LFCS-92-253
Hiding and Behaviour: An Institutional Approach
Rod Burstall and Răzvan Diaconescu

ECS-LFCS-93-254
Enrichment Through Variation
R. Gordon and A. J. Power

ECS-LFCS-93-255 (also published as CST-97-93)
A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification
Michael Mendler
PhD Thesis Price £8.50

ECS-LFCS-93-256
Statically Types Friendly Functions via Partially Abstract Types
B C Pierce and D N Turner

ECS-LFCS-93-257
Mistakes and Ambiguities in the Definition of Standard ML
Stefan Kahrs

ECS-LFCS-93-258 (also published as CST-98-93)
PAC-Learning Geometrical Figures
Paul W Goldberg
PhD Thesis Price £8.50

ECS-LFCS-93-259
Approximately Counting Hamilton Cycles in Dense Graphs
Martin Dyer, Alan Frieze and Mark Jerrum

ECS-LFCS-93-260
Simulated Annealing for Graph Bisection
Mark Jerrum and Gregory Sorkin

ECS-LFCS-93-261
Decidability Questions for Bisimilarity of Petri Nets and Some Related Problems
Petr Jančar

ECS-LFCS-93-262
Algebraic Theories for Name-Passing Calculi
Joachim Parrow and Davide Sangiorgi

ECS-LFCS-93-263
Literate Programming: A Review
Angus Duggan

ECS-LFCS-93-264
Action Structures for the π-Calculus
Robin Milner

except that axioms are allowed in module interface declarations (signatures) and in place of code in program modules (structures and functors). Some Extended ML specifications are executable, since Standard ML function definitions are just axioms of a certain special form. This makes Extended ML a "wide-spectrum" language which can be used to express every stage in the development of a Standard ML program from the initial high-level specification to the final program itself and including intermediate stages in which specification and program are intermingled.

Formally developing a program in Extended ML means writing a high-level specification of a generic Standard ML module and then refining this specification top-down by means of a sequence (actually, a tree) of development steps until an executable Standard ML program is obtained. The development has a tree-like structure since one of the ways to proceed from a specification is to decompose it into a number of smaller specifications which can then be independently refined further. In programming terms, this corresponds to implementing a program module by decomposing it into a number of independent sub-modules. The end-product is an interconnected collection of generic Standard ML modules, each with a complete and accurate specification of its interface with the rest of the system. The explicit interfaces enable correct re-use of the individual modules in other systems, and facilitate maintainability by making it possible to localize the effect on the system of subsequent changes in the requirements specification.

Current research is on: the semantics of Extended ML and the subtleties of the relationship between Extended ML and Standard ML; proving theorems in Extended ML specifications; tool support for specification and formal development; and case studies.

Don Sannella, Judith Underwood

Foundations for Formal Program Development

The research on practical formal development in Extended ML is complemented by research on algebraic and logical foundations for specification and formal program development. The work on foundations providesa solid basis for the practical work, giving substance to guarantees that a formally developed software system satisfies the specification of requirements from which the development process commenced. Also, simultaneous work on foundations and practice helps to ensure that the foundational work is largely inspired by practical concerns.

The most basic assumption of this work is that programs are modelled as many-sorted *algebras* or similar structures. This point of view abstracts from the concrete details of code and algorithms, and regards the input/output behaviour of functions and the representation of data as primary. Representing programs in terms of sets (of data values) and ordinary mathematical functions over these sets greatly simplifies the task of reasoning about program correctness.

Over the past decade or so, many of the components needed for a complete, general and coherent theory of specification and formal development of modular software systems have emerged. These include an understanding of: the structure of specifications and how this relates to the modular structure of programs; behavioural equivalence and its role in the implementation of data abstractions; stepwise refinement and implementation of specifications; parameterisation of specifications and program modules; proof in the context of structured specifications; and the degree to which such a theory can be made independent of the particular logical system used to write specifications. Much of this work has been done in the context of an algebraic specification language called ASL having a very simple semantics but which is nonetheless powerful enough to capture all the essential problems on a level at which their essence can easily be studied. Current research is on: term rewriting; various topics concerned with the extension from first-order to higher-order parameterisation;and relationships with type theory.

Don Sannella, Judith Underwood

Safety Critical Systems

Work in this area aims to employ foundational approaches to the analysis and design of Safety Critical Systems. Such are well suited to formal modelling and analysis. They are usually small, precisely (though informally) specified, and are developed by engineers who are open to any methods which might help to increase confidence in the safety of systems. This work has its origins in the "Mathematically Proven Safety Systems" (MPSS). During its lifetime MPSS worked with a variety of industrial organisation on anlysing and verifying safety properties of systems under development. Sectors in which work has been done includes nuclear protection, railway signalling, and civil aviation.

Building on experience gained from MPSS we have a new project "Communication in Safety Cases" which aims to look at safety from a number of perspectives, involving LFCS, the Departments of Sociology and Artificial Intelligence and the Human Communication Research Centre. The project is to explore communication failures in reasoning about safety and how mathematical methods can be deployed to analyse and ameliorate such failures. Important issues include: standards, their interaction, the integration of formal approaches to system dsign, and how all of this can be presented in a safety case which can be objectively assessed by certification authorities.

As work on safety develops, new activities have arisen. Through a Teaching Company project we will be working closely with safety consultants Adelard to develop industrial stregnth techniques for analysis and development of safety-critical systems. Work within the European Workshop on Industrial Computer Systems developing pre-standards allows the evaluation, exploitation and dissemination of our ideas. Collaboration with the Research Centre for Social Science and Sociology has seen the development of two funded projects on the application of formal

ECS-LFCS-92-242
Deliverables: a categorical approach to program development in type theory
Rod Burstall and James McKinna

ECS-LFCS-92-243
The Virtues of Eta-expansion
C Barry Jay and Neil Ghani

ECS-LFCS-92-244
Decomposability, Decidability and Axiomatisability for Bisimulation Equivalence on Basic Parallel Processes
S Christensen, Y Hirshfield and F Moller

ECS-LFCS-92-245
A Semantics for Static Type Inference
Gordon Plotkin

ECS-LFCS-92-246
Logic Programming via Proof-Valued Computations
David J Pym and Lincoln A Wallen

ECS-LFCS-92-247 (also published as CST-96-92)
Deliverables: a categorical approach to program development in type theory
James Hugh McKinna
PhD Thesis Price £9.00

ECS-LFCS-92-248
A Synopsis on the Identification of Linear Logic Programming Languages
James Harland and David Pym

ECS-LFCS-92-249
Action Structures
Robin Milner

ECS-LFCS-92-250 (UNAVAILABLE)
An Algorithm for Constructing $\beta\eta$-long normal forms
Philippa Gardner

ECS-LFCS-92-251
Equivalences between Logics and their Representing Type Theories
Philippa Gardner

ECS-LFCS-92-252
Newtonian Arbiters Cannot be Proven Correct
Michael Mendler & Terry Stroup

A. S. Wight et al. Performance evaluation of FDDI using simulation models. In *Proc. IEE Ninth UK Teletraffic Symposium*, page 6. IEE, 1992.

P. Yeung and D. J. Rees. Resources restricted aggressive scheduling. In *Proceedings of EDAC 92 Conference*, page 10, MAR 1992.

Reports & Theses 1992-93[5]

DEPARTMENTAL REPORTS

CSR-23-92
A Survey of PRAM Simulation Techniques
Tim J Harris

CSR-24-93
PEPA: Performance Enchanced Process Algebra
Jane Hillston

CSR-25-93
Parallel Programming: List Homomorphisms and the Maximum Segment Sum Problem
Murray Cole

CSR-26-93
Proceedings of Workshop on Process Algebra and Performance Modelling, University of Edinburgh, May 21, 1993
Editors: Jane Hillston and Faron Moller

CSR-27-93
The Performance of Barnes-Hut *N*-body Simulation in the Abstract
Todd Heywood, Kishan Mehrotra, Sanjay Ranka

CSR-28-93
Models of Parallelism
Todd Heywood and Claudia Leopold

LFCS Reports

ECS-LFCS-92-241 (also published as CST-95-92)
Correctness Proofs of Compilers and Debuggers: an Approach Based on Structural Operational Semantics
Fabio Q B da Silva
PhD Thesis Price £9.00

[5]Copies of reports and theses may be obtained by writing to Ms Lorraine Edgar at the Departmental address — by email: `lme@uk.ac.ed.dcs`

methods to safety systems and on the structure and organisation of high-integrity computing.

This work involves elemnts of technology transfer from basic theory to its application, thus it is important to develop industrial links. These include Shell Expro, British Rail, Lucas Automotive, Praxis, AEA Technologies and assessment authorities such as the UK HSE and the German TUV.

George Cleland, Glenn Bruns, Stuart Anderson, Terry Stroup

Formal and Mechanical Methods of Parallelisation

Programming should be a problem solving activity. Issues that have nothing to do with the problem should, if at all possible, not be the burden of the programmer but should be part of the compilation process. In many applications, parallelism and communication are implementation, not problem solving issues. In this sense, they are 'low-level' concepts like gotos or pointers, except that they are more difficult to use and verify.

There are classes of programs into which maximum parallelism can be infused mechanically. The resulting parallel program emulates a *systolic array*. A systolic array is a distributed network of sequential processors that are linked together by channels in a particularly regular structure. Typical applications are highly repetitive algorithms on large data structures such as those which occur in image or signal processing, meteorology, etc.

Automatic methods distribute the operations of source programs that do not specify concurrency or communication into time and space (*systolic design*). The target descriptions of these methods are distribution functions that form an abstract specification of the systolic array. The array can then be refined into software (i.e. into a distributed program) by a process of *systolizing compilation* or into hardware (i.e. into a chip layout.

Current research includes (1) the systolization of application problems, (2) classifications and extensions of systolic design methods and (3) the development of systolizing compilation techniques. An implementation with graphics facilities of a method that transforms imperative programs into systolic arrays is in use and systolic programs are being run on the Meiko Computing Surface in EPCC.

Damal Arvind, Brian McConnell, Jonathan Knight

Languages and Applied Semantics

Standard ML

ML has evolved over a period of about fifteen years. Since 1985 there has been a major effort at Edinburgh to give a precise definition of the syntax and semantics of the language, both of the core language and of the modules language,

which is based on David MacQueen's original modules proposal. This work is now completed and in 1987 Robin Milner and Rod Burstall were awarded the BCS Technical award for 'Standard ML'.

Currently, research is focussed on integrating concurrency and ML. Several implementations of ML have included experimental features for creating processes and passing values between processes. However, none of these have been given a formal semantic definition. The ML team has developed a simple, but expressive, description for a powerful set of concurrent operations, and are developing proof techniques to support reasoning about this definition.

Having developed a concurrent extension of ML on a shared memory system (based upon the POLY/ML system), the ML team is now working on a support for Standard ML on distributed memory systems, such as networks of workstations or transputer arrays. Part of this work involves proving that the synchronisation algorithm implements the formal semantics correctly.

The ML team is also exploring ways of defining its concurrent extension of ML in terms of the Π-calculus of Milner, Parrow and Walker, which provides a common framework for functions and communicating processes. This approach could yield a semantics that better supports formal reasoning about concurrent programs.

A Support Tool for Operational Semantics

The experience of defining Standard ML in terms of operational semantics highlighted the need for tools to manipulate and reason about semantic descriptions. An attempt is being made to encode the operational semantics formalism as a logic in theorem provers such as HOL or Lego. This will be augmented with a front end that will let the language designer enter and manipulate semantic rules in their usual syntax. Proof tactics will be developed to support reasoning about large examples in this system, with the eventual aim being to handle large parts of the definition of Standard ML.

Kevin Mitchell, Renaud Marlet

Categorical Semantics

A categorical semantics for a programming language interprets programs by the morphisms of a category. Current research at the University of Edinburgh (funded by BP and The Royal Society of Edinburgh) considers interpretations in *ordered* categories, so that programs having the same denotation (or satisfying the same specification) can be ordered according to their ancillary properties, e.g. reduction order (in a rewrite system), resource use, degree of determinism. The intention is to combine the operational flavour of the order with the powerful mathematical understanding of the usual unordered semantics.

A. J. Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. In *Proc. 1st Latin American Symposium on Theoretical Informatitcs*, volume 583, page 14. Springer Verlag, APR 1992.

A. J. Sinclair. Improved bounds on mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:351–370, DEC 1992.

A. J. Sinclair, M. R. Jerrum, and B. McKay. When is a graphical sequence stable? In *Random Graphs*, volume 2, page 15. Wiley, 1992.

A. J. Sinclair, Y. Rabinovich, and A. Wigderson. Quadratic dynamical systems. In *Proc. 33rd Symposium on Foundations of Computer Science*, page 10. IEEE, OCT 1992.

B. U. Steffen, C. B. Jay, and M. Mendler. Compositional characterization of observable program properties. *Theoretical Informatics and Applications*, 26(5), 1992.

C. P. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science*, volume 2, pages 477–563. Oxford University Press, DEC 1992.

C. P. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science*, volume 2, pages 477–563. Oxford University Press, DEC 1992.

R. D. Tennent. Denotational semantics. In *Handbook of Logic in Computer Science (Editors: S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum)*. Oxford University Press, 1992.

R. D. Tennent et al. Semantical analysis of specification logic, part 2. *Information and Computation*, 1992.

P. Thanisch et al. Compact scheme forests in nested normal form. *International Journal of Computer Mathematics*, 45:23–48, 1992.

P. Thanisch et al. Finding compact scheme forests in nested normal form is NP-hard. *Information and Computation*, 1992.

N. P. Topham et al. A comparison of two memory models for high performance computers. In *Proceedings of CONPAR/VAPP-92, Lyon*, 1992.

N. P. Topham et al. Semantics driven computer architecture. In *Proc. Parallel Computing 1991*, page 8. Elsevier Science Publishers, 1992.

L. Wang. Deriving a correct computer. In *Proceedings of the 1992 International Workshop on Higher Order Logic Theorem Proving and its Applications*, SEP 1992.

L. Wang. Formal derivation of a computer. In *Correct Hardware Design Methodologies*. Elsevier, 1992.

A. S. Wight. What does benchmarking mean today? In *8th UK Performance Engineering Workshop*, Sep 1992.

G. D. M. Ross. Managinx x in a large distributed environment. In *Proceedings of the Fourth Annual Conference*, page 7. European X User Group, SEP 1992.

D. Sangiorgi. Barbed bisimulation. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623, pages 685–695. Springer-Verlag, 1992.

D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *Proceedings of seventh annual IEEE Symposium on Logic in Computer Science (LICS'92)*, pages 102–109. IEEE Computer Society Press, 1992.

D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *Proceedings of seventh annual IEEE Symposium on Logic in Computer Science (LICS '92)*, pages 102–109. IEEE Computer Society Press, 1992.

D. Sangiorgi. The problem of weak bisimulation up to. In *Third International Conference on Concurrency Theory (CONCUR'92)*, volume 630, pages 32–46. Springer-Verlag, 1992.

D. Sangiorgi et al. Barbed bisimulation. In *Proceedings of 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, volume 623, pages 685–695. Springer-Verlag, 1992.

D. Sangiorgi and A. J. R. G. Milner. The problem of weak bisimulation up to. In *Third International Conference on Concurrency Theory (CONCUR '92)*, volume 630, pages 32–46. Springer-Verlag, 1992.

D. T. Sannella, S. Sokolowski, and A. Tarlecki. Toward formal development of programs from algebraic specifications: parameterisation revisited. *?*, 29:689–736, DEC 1992.

D. T. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: model-theoretic foundations. In *Proc. Intl. Colloq. on Automata, Languages and Programming*, page 16. Springer Verlag, 1992.

D. T. Sannella, A. Tarlecki, and S. Sokolowski. Toward formal development of programs from algebraic specifications: Parameterisation revisited. *?*, 29:689–736, DEC 1992.

D. T. Sannella and L. A. Wallen. A calculus for the construction of modular prolog programs. *Journal of Logic Programming*, 12:147–177, 1992.

A. K. Simpson et al. Hierarchical meta-logics. In *Proceedings of META-92*, page 13, JUN 1992.

A. J. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhäuser Boston, DEC 1992.

A. J. Sinclair. Improved bounds for mising rates of markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:20, DEC 1992.

Other recent work uses elementary limits and colimits to provide clean definitions and manipulations of datatypes. Examples include the use of pullbacks and express side-conditions, and invariants of loops to interpret tail recursion.

Barry Jay

Axiomatic and Categorical Models for ML

It is important that new users can be introduced to ML without having to master the technicalities of the formal semantics. An attempt is currently being made to formalise naive models of ML which provide a sound basis for working with portions of the language.

Mike Fourman, Wesley Phoa

Semantics

Research in this area comprises work on the mathematical foundations of the semantics of computationally-oriented languages and the investigation of particular areas which are not the focus of any large research project. One needs to understand the operational and denotational semantics of languages, and logics to reason with them. The aims are, for example, to pursue language design (whether for hardware, programming or specification) and program and specification design and to prove systems meet requirements. The mathematical tools available are varied and now quite sophisticated and are taken from logic, lattice theory, topology, and algebra, and especially category theory which provides a rich source of unifying language.

Gordon Plotkin, John Power

Logics and Proof

Lego: Interactive Proof System for Mathematics and Software Development

We have developed an interactive proof editor, LEGO, based upon extensions of Coquand and Huet's Calculus of Constructions. The extensions include a cumulative hierarchy of universes and inductive types. LEGO is a pragmatic implementation, and includes syntactic and information management features, intended to support large formal developments. We are exploring applications to mathematics and software development. Recent examples include the strong normalisation of the second-order λ-calculus, Lagrange's theorem in number theory and a large metatheory of Pure Type Systems including λ-calculus, type theory and typechecking algorithms similar to those used in LEGO itself. Work on formal

program development has drawn ideas from category theory and from research on data refinement and algebraic specification.

Randy Pollack, Rod Burstall, Zhaohui Luo, James McKinna

Logical Frameworks

The aim of the Logical Frameworks project is to study theories of formal reasoning, addressing such questions as: What is a logical system? What is a presentation of a logical system? These questions (and their answers) are motivated, in particular, by the requirements of mechanical implementation of formal reasoning systems.

One approach to these issues is to consider a type-theoretic metalanguage, which can be implemented on a machine, in which logical systems can be represented by utilizing encoding paradigms based upon the Curry-Howard identification of formulae and types. This is the approach taken by the Edinburgh Logical Framework – ELF. An alternative approach is to implement a particular, powerful logic. This approach can also exploit type theory, an example being Huet and Coquand's Calculus of Constructions. Recent work in these areas concerns the exploitation of Generalized Type Systems as metalanguages for Logical Frameworks. It remains a question of active research to determine which are the better type theories to use as logical frameworks. An alternative approach is provided by studying algebraic (categorical) presentations of type theory. In a similar vein it is hoped that functional calculi, in the manner of Kelly and Hagino, might provide a language for the definition of logical systems in which the basic structural properties of logical systems, in particular for Sequent calculi, can be studied.

Gordon Plotkin, John Power, Phillippa Gardner, Zhaohui Luo, Christoph Raffalli, David Pym

Parallel Computing

Parallelization Tools

Parallelization tools form essential components of the support environment on parallel MIMD computing systems, but on many systems these tools are still either non-existent or relatively unsophisticated. Parallelisation tools can be categorised as either synthesis tools or analysis tools. Synthesis tools may be interactive or automatic, and both categories are being investigated. It is expected that the design of automatic tools will benefit considerably from experience gained with the use of the interactive tools. Thus the design of parallelising compilers will benefit from experience gained with code browsers, and automatic process/data mapping tools from experience with process/data placement aids.

The analysis tools will gather data during the running of programs and provide various types of measurement relating to the performance of programs and the

J. Power and C. Wells. A formalism for the specification of essentially-algebraic structures in 2-categories. *International Journal on the Foundations of Computer Science*, 2:1–28, 1992.

R. N. Procter. Applications of groupware in the financial sector. *Financial Technology Insight*, page 5, NOV 1992.

R. N. Procter. Groupware — IT's new wave? *Financial Technology Insight*, pages 8–12, AUG 1992.

R. N. Procter. Human factors in financial services IT. *Financial Technology Insight*, pages 8–11, APR 1992.

R. N. Procter. Information technology in the financial services sector. In *Strategic Change and IT in the Retail Financial Service*, page 22. Prentice Hall, 1992.

R. N. Procter. Object-oriented software design. *Financial Technology Insight*, pages 9–12, MAY 1992.

R. N. Procter. Promoting the usability of IT systems. *Financial Technology Insight*, pages 7–8, MAR 1992.

R. N. Procter et al. HCI: Whose problem is IT anyway? In *Proceedings of the 5th IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, pages 385–396. North-Holland, AUG 1992.

R. N. Procter, Scarbrough, Williams, Fincham, Fleck, and Tierney. Expertise and innovation: It strategies in the financial services sector. In *Proceedings of the Edinburgh PICT Workshop on Exploring Expertise in Technological Innovation*. Edinburgh PICT Centre, NOV 1992.

R. N. Procter and Williams. Hci: Whose problem is it anyway? In *Proceedings of the CRICT Conference on Software and Systems Practice: Social Science Perspectives*, page 12. Centre for Research Into Innovation, Culture and Technology (CRICT), DEC 1992.

Y. Rabinovich, A. J. Sinclair, and A. Wigderson. Quadratic dynamical systems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 304–313. IEEE, OCT 1992.

M. Read and E. Kazmierczak. Formal program development in modular prolog: A case study. In *LOPSTR'91*, page 23. Springer Verlag, 1992.

T. Reinhardt, M. Mendler, and R. T. Stroup. Die formale validierung einer "bausteintafel" delay-insensitiver grundlemente. In *Proc. ITG/GME/GI Fachtagung Rechnergestützter Entwurf und Architektur mikroelektronischer Systeme, Poster Session, Darmstadt*, NOV 1992.

G. D. M. Ross. Managing x in a large distributed environment. In *Proceedings of the 4th Annual Conference of the European X User Group*, pages 203–209. European X User Group, SEP 1992.

G. D. Plotkin. Editor. *Information and Computation*, 1992.

G. D. Plotkin. Editor. *Mathematical Structures in Computer Science*, 1992.

G. D. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Typed Lambda Calculi and Applications*, page 15. Springer-Verlag, 1992.

G. D. Plotkin and M. Abadi. A logical view of composition. ?, 114(1):28, 1992.

G. D. Plotkin, M. Abadi, M. Burrows, and B. Lampson. A calculus for access control in distributed systems. In *Advances in Cryptology-Crypto'91*, volume 576, page 23. Springer-Verlag, 1992.

G. D. Plotkin et al. A logical view of composition. *Theoretical Computer Science*, 1992.

G. D. Plotkin, R. Harper, and F. Honsell. A framework for defining logics. *Journal of the Association of Computing Machinery*, 40(1):41, 1992.

R. A. Pollack. Typechecking in pure type systems. In *Proceedings LF Workshop, Båstad*, 1992.

R. J. Pooley. DEMOGRAPHER, a tool for generation of DEMOS simulation and CCS process algebra models from graphical descriptions using x-windows, JUN 1992.

R. J. Pooley. DEMOGRAPHER PC, a tool for generation of DEMOS simulation and CCS process algebra models from graphical descriptions using VGA graphics, JUN 1992.

R. J. Pooley. Deriving the functional properties of process based simulation models. In *Proceedings of the EUROSIM '92 Simulation Congress*, pages 229–234. Elsevier, SEP 1992.

R. J. Pooley. Formalising the description of process based simulation models. In *Proceedings of the 24th Summer Computer Simulation Conference*, pages 93–97. Simulation Councils Inc, JUL 1992.

R. J. Pooley. Process algebras and performance modelling. In *Proceedings of the 8th UK Computer and Telecommunications Performance Engineering Workshop*, pages 86–95. BCS Performance Engineering Specialist Group, SEP 1992.

R. J. Pooley. Review of "performance engineering of software systems" by connie u.smith. *Performance Evaluation Review*, pages 23–24, NOV 1992.

R. J. Pooley et al. The derivation of functional properties of process based simulation models. In *Proceedings of the EUROSIM Simulation Congress*, page 12. Elsevier, JUN 1992.

R. J. Pooley and A. L. Song. An editing and checking tool for stochastic petri nets. In *Modelling and Simulation 1992*, pages 228–232. Society for Computer Simulation, Europe, JUN 1992.

details of run-time activity within them. Such measurements will provide feedback on the efficacy of the synthesis tools and will also lead to a better understanding of parallel computation generally.

Rosemary Candlin

Skeletal Parallel Programming

Many efficient algorithms for message-passing parallel computers share common patterns of data distribution, process distribution and communication. The complexity of programming such machines can be reduced (for some problems) by defining a library of such patterns of computation, expressed imperatively as equivalent sequential program skeletons, or declaratively as higher order functions. Work in this area involves the collection of a set of such structures and an investigation of the implementation decisions involved when a program is constructed as a complex composition of several skeleton instances. When presented in a functional context (for example as in the Bird-Meertens theory of lists) interesting opportunites for program devlopment by transformation arise.

Murray Cole

The Dynamic Behaviour of Parallel Programs

The aim of this work is to discover whether there are similarities, from a statistical point of view, in the run-time behaviour of various types of parallel programs, so that operating systems could be designed which would take over the burden of process allocation from the applications programmer. Unlike most performance studies of parallel programs, where the main interest has been to minimize the execution times of particular programs by an investigation of the detailed instruction sequences, this work is directed towards characterizing *classes* of parallel program in terms of a small number of parameters which refer to global properties of the program, like average process granularity, or average message size. This approach works well for one of the simplest (and commonest) types of parallel program with a fixed number of communicating processes. Good, quantitative predictions of performance in a given environment can be obtained from empirically-determined formulae. It is also possible to relate the improvement that can be brought about by run-time process migration to the parameter values of the program model. Currently, this work is being extended in two directions: to a study of models which are appropriate for programs with a changing behaviour at run-time, and to the characterization of complete computing systems, consisting of machine and systems programs as well as user programs.

These studies are largely based on simulation experiments with random programs, but confirmation that the results are applicable in practice comes from experimental work on the Meiko Computing Surface and on a special-purpose multiprocessor machine in the department. This machine is designed as a testbed

for experiments on parallel program performance: it has hardware monitoring facilities and permits very detailed measurement of operating systems and user program costs to be made in a non-intrusive manner. The feasibility of supporting process migration transparently and efficiently has been demonstrated by the construction of an operating system for this machine.

Rosemary Candlin

Models of Parallelism

A *model* of parallelism is an abstract view of a parallel computing system, or more appropriately a part of a system, obtained by removing details in order to allow one to discover and work with the basic principles. Within the context of a hierarchy of model types at different levels of abstraction - architectural, computational and programming models - research interests include the definitions of particular models and the mappings between different model types. The aim is to find a means of bridging the gap between theory and practice of parallel computing. Particularly desired system principles are cost-effectiveness and scalability.

The Hierarchical PRAM (H-PRAM) model is the main focus of work into computational models. Based on this model, work in progress includes parallel algorithm design and analysis, investigation into its relationships to overlying programming models based on Algorithmic Skeletons, and mapping of the model to parallel architectures as simulated by the department's HASE architecture simulation environment.

Todd Heywood

Performance Modelling

The emergence of practical distributed and parallel computer systems is generating new classes of problem, which are not solvable by existing analytic and numerical methods and are driving the search for new solution methods and more importantly focussing attention again on simulation programming. Also the arrival of powerful and inexpensive workstations with good graphics capabilities makes it worthwhile to explore the construction of novel tools to support simulation and analysis.

Currently much of the research is aimed at combining traditional performance modelling and work in concurrency. This uses process algebras to express properties of stochastic models, including stochastic Petri nets and simulation models.

Work in the application of performance modelling includes communication networks, distributed systems and novel architectures.

Rob Pooley

M. Mendler and R. T. Stroup. Newtonian arbiters cannot be proven correct. In *Proc. Workshop on Designing Correct Circuits (Eds J. Staunstrup and R. Sharp)*. North-Holland, JAN 1992.

A. J. R. G. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

A. J. R. G. Milner and K. Larsen. A compositional protocol verification using relativized bisimulation. *Information and Computation*, 1:80–108, 1992.

A. J. R. G. Milner, J. Parrow, and D. J. Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1):1–40, 1992.

A. J. R. G. Milner, J. Parrow, and D. J. Walker. A calculus of mobile processes, part ii. *Information and Computation*, 100(1):41–77, 1992.

F. G. Moller, S. Christensen, and Y. Hirshfeld. Context-free processes and beyond: The decidability of infinite-state systems. In *Proceedings of ERCIM Workshop*. University of Pisa, DEC 1992.

F. G. Moller and C. Tofts. Behavioural abstraction in TCCS. In *Proceedings of 19th ICALP*, page 12. Springer-Verlag, JUL 1992.

F. G. Moller and C. Tofts. An overview of TCCS. In *Proceedings of Euromicro '92*, page 6, JUN 1992.

P. Moscato and M. Norman. A memetic approach for the travelling salesman problem implementation of a computational ecology for combinatorial optimisation on message passing systems. In *Parallel Computing and Transputer Applications*. IOS Press, 1992.

M. Norman. Bulk synchronous parallelism. *Wotug Newsletter*, 17, Jul 1992.

M. Norman, P. Thanisch, and K. Chang. Partitioning DAG computations: a cautionary note. In *Parallel Computing: From Theory to Sound Practice (Eds W. Joosen and E. Milgrom)*, pages 360–363. IOS Press, 1992.

O'Hearn and R. D. Tennent. Semantics of local variables. In *Proceedings of the London Mathematical Society Symposium on Applications of Categories in Computer Science, Durham, England, July 20-30, 1991*, volume 177. Cambridge University Press, 1992.

S. Pelagatti and S. Antonelli. On the complexity of the mapping problem for massively parallel architectures. *International Journal of Foundations of Computer Science*, pages 379–387, 1992.

J. G. Phillips and E. R. S. Candlin. An environnment for investigating the effectiveness of process migration strategies on transputer-based machines. In *Transputer Systems - Ongoing Research (Proceedings World Occam and Transputer User Group Technical meeting 15)*, pages 13–23. IOS Press (Amsterdam), APR 1992.

B. C. Pierce. F-omega-sub user's manual, version 1.2, OCT 1992.

M. R. Jerrum. Editor. *Combinatorics, Probability and Computing*, 1992.

M. R. Jerrum. Large cliques elude the metropolis process. *Random Structures and Algorithms*, 3(4):347–359, 1992.

M. R. Jerrum. Review of "probabilistic analysis of packing and partitioning algorithms" by e.coffman jr and g.lueker. *The Annals of Probability*, 20(4):2164–2167, 1992.

M. R. Jerrum, B. McKay, and A. J. Sinclair. When is a graphical sequence stable? In *Random Graphs (A.Frieze and T.Luczak, eds)*, volume 2, pages 101–115. Wiley, 1992.

M. R. Jerrum and U. Vazirani. A mildly exponential approximation algorithm for the permanent. In *Proceedings of the 33rd Annual IEEE Conference on Foundations of Computer Science*, pages 320–326. IEEE Computer Society Press, OCT 1992.

S. M. Kahrs. Context rewriting. In *Proceedings CTRS'92*, volume 656, pages 21–35. Springer-Verlag, JUL 1992.

S. M. Kahrs. Polymorphic type checking by interpretation of code. In *Proceedings of the 4th International Workshop on the Parallel Implementation of Functional Languages*, pages 59–70. RWTH Aachen, 1992.

S. M. Kahrs. Unlimp, uniqueness as a leitmotiv for implementation. In *Proceedings PLILP'92*, volume 631, pages 115–129. Springer-Verlag, AUG 1992.

K. Kalorkoti. On inverting polynomials and formal power series. *SIAM Journal on Computing*, page 9, 1992.

E. Kazmierczak. Modularizing the specification of a small database system in extended ML. *Formal Aspects of Computing*, 4(1):42, 1992.

C. Lengauer and D. Sangiorgi. An improved systolic array for string correction. In *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies*, pages 205–218. Elsevier, 1992.

C. Lengauer and J. Xue. Specifying control signals for one-dimensional systolic arrays by uniform recurrence equations. In *Proc. Algorithms and Parallel VLSI Architectures II*. Elsevier, 1992.

Z. Luo. A unifying theory of dependent types: the schematic approach. In *Proc. of Symp. on Logical Foundations of Computer Science (Logic at Tver)*. Springer-Verlag, 1992.

P. Martin and S. D. Gilmore. Pragmatic experience of the formal specification of a distributed operating system. In *Proceedings of the 5th International Conference on 'Putting into Practice Methods and Tools for Information System Design'*, pages 31–43, SEP 1992.

Performance Modelling

The emergence of practical distributed and parallel computer systems is generating new classes of problem, which are not solvable by existing analytic and numerical methods and are driving the search for new solution methods and more importantly focussing attention again on simulation programming. Also the arrival of powerful and inexpensive workstations with good graphics capabilities makes it worthwhile to explore the construction of novel tools to support simulation and analysis.

Currently much of the research is aimed at combining traditional performance modelling and work in concurrency. This uses process algebras to express properties of stochastic models, including stochastic Petri nets and simulation models.

Work in the application of performance modelling includes communication networks, distributed systems and novel architectures.

Rob Pooley, Alex Wight

Very Large Scale Integration

The main themes of activity are Computer Aided Design (CAD) of VLSI circuits and novel VLSI Architectures. CAD tools being developed include those involving the use of formal techniques for synthesis and verification of designs. Behavioural (high-level) synthesis is also under investigation.

Formal Techniques

Current CAD tools for VLSI manage complexity by manipulating a structural hierarchy. To manage the increasing complexity resulting from technological advances of VLSI fabrication, tools are required which can manage the hierarchy of behavioural abstractions used in system design. A formal model which naturally and accurately represents circuit behaviour is central to such tools.

Behavioural models with the generality necessary to describe behaviour at many levels have been introduced by researchers studying formal verification of hardware designs. In this work, techniques from formal logic have been used to establish the correctness of a circuit design by mathematical proof prior to fabrication.

The major thrust of current research at Edinburgh involves the development, in cooperation with industry, of practical formal models of digital behaviour, and of the temporal and data abstractions employed in system design. This work is applied to the development of sound design methodologies and to the development of design tools for interactive and automated behavioural synthesis and verification of digital systems, both hardware and software. These tools will provide high-level interfaces to the silicon compilers described above.

Mike Fourman, Antony McIsacc

Novel Architectures

A cellular array structure has been developed in which the array elements are simple but configurable logic blocks. Not only is the function of each element configurable, but also its communication with adjacent elements. The increasing density of fabrication technologies means that increasingly large arrays of elements can be fabricated, and so realistically-sized computations can be mapped onto the arrays with large speed gains resulting from concurrency. Various applications of arrays are being investigated, including the design of program-specific arithmetic pipelines, the emulation of hypercube algorithms and the implementation of data compression and encryption algorithms.

Gordon Brebner, David Rees

Research Units

Two areas of the work of the department have developed to the stage that they have the status of research units. The Laboratory for Foundations of Computer Science (LFCS) is wholly within the department. Established in 1986, it has a world-wide repuatation for work in the foundations of Computer Science. LFCS supports a large international community of researchers, visitors and PhD students and has strong links with European and North American research groups. Edinburgh Parallel Computing Centre (EPCC) is a more recently established unit (???) and is a joint organisation between our department and the Department of Physics. EPCC has a growing international repuation and has a strong programme of industrial application projects.

The Edinburgh Parallel Computing Centre

The University of Edinburgh has a long history of involvement in parallel computing and is one of the largest centres of research in the theory and application of parallelism in Europe. Work started in the early 1980s with the use of ICL Distributed Array Processors (DAPs) by the Departments of Physics and Molecular Biology. When a Meiko Computing Surface was acquired in 1986, the Edinburgh Concurrent Supercomputer Project was set up. In 1990, as a way of bringing more closely together the many different aspects of parallel computing research being carried out in the University, the Department of Physics entered into a collaborative venture with the Department of Computer Science and the Edinburgh University Computing Service to establish the Edinburgh Parallel Computing Centre (EPCC). Its aim is to accelerate the effective exploitation of high performance parallel computing systems throughout academia, industry and commerce.

The Centre is interdisciplinary and acts as a focus for work being carried out within a number of departments. It has expanded rapidly since 1990, and now has a staff of over 45. The majority of staff are employed on collaborative projects

S. D. Gilmore. Book review: Programming with VDM. *University Computing*, 1992.

S. D. Gilmore and M. Clint. Evaluating formal methods. In *Proceedings of the 5th International Conference on 'Putting into Practice Methods and Tools for Information System Design'*, pages 15–29, SEP 1992.

H. Goguen and Z. Luo. Inductive data types: Well-ordering types revisited. In *Logical Frameworks*, page 19. Cambridge University Press, 1992.

L. Goldberg. Automating polya theory: The computational complexity of the cycle index polynomial. *Information and Computation*, 1992.

L. Goldberg. Efficient algorithms for listing unlabeled graphs. *Journal of Algorithms*, 13(1):128–143, 1992.

K. Goossens. Integrating hardware description languages and proof systems. In *Poster presentation at IFIP '92, Madrid*, Sep 1992.

K. Goossens. Operational semantics based formal symbolic simulation. In *Higher Order Logic Theorem Proving and Its Applications, Leuven, Belgium. (Eds: L. Claesen and Michael Gordon)*, pages 487–506. North Holland, SEP 1992.

J. Groote and F. G. Moller. Verification of parallel systems via decomposition. In *Proceedings of CONCUR '92*. Springer-Verlag, AUG 1992.

R. Harper, F. Honsell, and G. D. Plotkin. A framework for defining logics. *Journal of the Association of Computing Machinery*, 1992.

T. H. Heywood et al. A practical hierarchical model of parallel computation i: The model. *Journal of Parallel and Distributed Computing*, NOV 1992.

T. H. Heywood et al. A practical hierarchical model of parallel computation ii: Binary tree and fft graph algorithms. *Journal of Parallel and Distributed Computing*, NOV 1992.

T. H. Heywood et al. Sorting and list ranking on the hierarchical PRAM model. In *4th Symposium on the Frontiers of Massively Parallel Computation*, OCT 1992.

J. Hillston. A tool to enhance model exploitation. In *Computer Performance Evaluation '92: Proceedings of the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 179–193. Edinburgh University Press, SEP 1992.

R. N. Ibbett. Parallel computer architectures: Where next? In *Proc. Transputers'92*, page 20. IOS, May 1992.

R. N. Ibbett and D. J. Rogers. Xbar: a VLSI circuit for bit-sliced packet switching networks. In *Proceedings of IFIP CONGRESS '92*, volume 1, pages 562–570. Elsever Science Publishers, SEP 1992.

R. N. Ibbett, N. P. Topham, et al. *Editor of Programming Environments for Parallel Computing*. Elsevier Science Publishers, 1992.

G. Bruns and S. O. Anderson. A case study in the analysis of safety requirements. In *Proceedings Safecomp '92*. Pergamom Press, 1992.

R. Burstall and J. Goguen. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, JAN 1992.

R. Burstall and J. McKinna. Deliverable: a categorical approach to program development in type theory. In *Proceedings of the Third Annual BRA Workshop on Types for Proofs and Programs*, MAY 1992.

E. R. S. Candlin, P. J. Fisk, J. G. Phillips, and N. Skilling. A statistical approach to predicting the performance of concurrent programs. In *Parallel Computing: From Theory to Sound Practice (Proceedings of the European Workshop on Parallel Computing, Barcelona)*, pages 104–107. IOS Press (Amsterdam), MAR 1992.

N. Carmichael and M. Norman. Parallel processing: the power and the portability. experiments with 'reusable toolkits'. *Future Generations Computer Systems*, 8(1):3–8, JUL 1992.

L. Chen. An interleaving model for real-time systems. In *Symposium of Logical Foundations of Computer Science - Tver'92*, volume 620, pages 81–92. Springer Verlag, 1992.

S. Christensen, Y. Hirshfeld, and F. G. Moller. Context-free processes and beyond: The decidability of infinite-state systems. In *Proceedings of ERCIM Workshop*. University of Pisa, DEC 1992.

S. Christensen, H. Huttel, and C. P. Stirling. Bisimulation equivalence is decidable for all context-free processes. In *Proceedings of CONCUR 92*, volume 458, pages 138–147. Springer-Verlag, AUG 1992.

G. L. Cleland. ... and the third pig built his house of bricks. In *SERC/DTI Joint Framework for Information Technology Technical Conference*, MAR 1992.

M. I. Cole. Parallel software paradigms. In *Advances in Parallel Algorithms*, chapter 1, page 25. Blackwell Scientific Publications, 1992.

M. I. Cole. Writing parallel programs in non-parallel languages. In *Software for Parallel Computers*, pages 315–323. Chapman & Hall, 1992.

M. F. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. In *Proceedings of the 17th Colloquium on Trees in Algebra and Programming*, volume 581, pages 145–164. Springer-Verlag, FEB 1992.

M. F. Dam. R-generability, and definability in branching time logics. *Information Processing Letters*, 41:281–287, 1992.

M. P. Fourman. The CAD tool of the future. In *IFIP '92 Proceedings of the 12th World Computer Congress Madrid, Spain*, volume 1. Elsevier, 1992.

S. D. Gilmore. Book review: Action semantics. *University Computing*, 1992.

with industry to investigate the suitability of parallel computing for a range of applications. It also provides service and support for users on a number of parallel machines including the Edinburgh Concurrent Supercomputer, an AMT DAP, and the two most powerful parallel machines in the UK: a CM-200 from Thinking Machines Corporation of Cambridge, Mass., and a Meiko i860 Computing Surface. All the machines can be accessed by both local and remote users over JANET.

An increasing amount of EPCC's work is being done through industrial and commercial contracts and collaborations. EPCC is one of four UK centres taking part in the DTI/SERC Parallel Applications Programme (PAP), which aims to bring together pre-eminent centres for parallel processing, industrial and commercial end-users, system vendors and software houses, to produce applications and tools. The EPCC PAP contract, itself worth £3.5 million, involves attracting industrial sponsorship to bring the total value of the project to £9 million over four years.

Organisation and Management

In addition to direct industrial support, EPCC is supported by major grants and contracts from the Advisory Board for the Research Councils, the Department of Trade and Industry, the Informations Systems Committee of the Universities Funding Council, the Science and Engineering Research Council, Scottish Enterprise Software Group and the Commission of the European Communities. It is run under the auspices of an Advisory Board made up of representatives from the funding bodies, the University, industry and users.

The work of the Centre is overseen by the Director, (Professor D.J.Wallace of Physics)[2], who works in consultation with an Executive Committee which includes the Chairman (Professor J. Collins, formerly of Electrical Engineering), the Assistant Director (Professor R.N. Ibbett of Computer Science), the Director of the EUCS and the Commercial and General Managers of EPCC. The Committee meets regularly to review the work of the Centre, resource allocation, industrial involvement, and bids for future funding.

Other groups within the University contribute to the work of the Centre and are able to offer a variety of additional facilities to members of the Centre. Membership of the Centre is open to all staff and research students whose work relates to the interests of the Centre. Many members of the Department of Computer Science are also members of the Centre, and a variety of student projects at both undergraduate and postgraduate level involve use of EPCC facilities. The Centre also has a flourishing Summer Student Scholarship Scheme, currently involving some 20 students per annum, in which Computer Science undergraduates frequently participate.

[2]Professor Wallace has been appointed Vice-Chancellor of Loughborough University, with effect from 1 October 1993; his EPCC rôle will be taken over by by Dr Richard Kenway, Department of Physics

Laboratory for the Foundations of Computer Science (LFCS)

The Laboratory (LFCS) has a long record of seminal research into the Foundations of Computer Science. This includes research into the fundamental theory, the construction of prototype systems and methods which embody that theory and the development of applied science.

This work has strongly influenced the development of Computation Theory particularly in the fields of semantics, specification, concurrency, complexity machine assisted proof and implementations (both of languages and of proof systems). It is naturally the software products which have been most visible, in particular the functional programming language Standard ML, which is now in use in many locations around the world both in educational and in research and development environments. Methodologies have also been exported; examples are the Specification Language CLEAR and the algebraic Calculus of Communicating Systems.

From these roots the Laboratory was formed in 1986 with the twin aims of intensifying the basic research and deepening formal links with industry to ensure the application of this work in practical environments. Significant funding from the Science and Engineering Research Council, national and international informatics programmes, and Industry established the Laboratory and strengthened its research programme. After its initial period of operation it is approaching its planned size and is generally accepted as one of the leading centres in the world in Theoretical Computer Science and its application.

In recent years there has been significant expansion of applications and case study work. This is typified by work in safety-critical systems. LFCS is the focus of a project across four departments in the University which is studying the semantic basis of safety systems and their associated safety case. Applications work with industry continues to develop. We have strong links with many companies, both large and small. Particularly strong links exist with Adelard, Harlequin, DEC (UK & US), Praxis, Abstract Hardware and Shell.

LFCS is coordinating node for the EC funded Network of Excellence in the Logical Foundations of Computer Science – EuroFOCS. The aim of the Network, established in October 1993 is to develop a coherent European research culture in its cognate area, and to promote awareness across Europe.

LFCS Structure and Management

The Laboratory is a research unit of the University, but contained within the department. It has currently about 70 members: 14 members of teaching staff, 23 research fellows, 6 technical and administrative support and about 30 PhD students. There are a number of other members of the University who are associate members of the Laboratory (from the Artificial Intelligence Department and the Centre for Cognitive Science).

The Laboratory is managed on a day to day basis by its Directorate: Prof.

Publications

The department publishes widely and has a thriving series of internal reports the following sections summarise this year's work.

Publications

M. Abadi, M. Burrows, B. Lampson, and G. D. Plotkin. A calculus for access control in distributed systems. In *Advances in Cryptography - Crypto '91 (Ed: J. Feigenbaum)*, volume 576, pages 1–23. Springer-Verlag, 1992.

M. Abadi, L. Cardelli, B. C. Pierce, and D. Rémy. Dynamic typing in polymorphic languages. In *Proceedings of the ACM SIGPLAN Workshop on ML and its Applications*. ACM, JUN 1992.

P. Anderson. Effective use of local workstation disks in an NFS network. In *LISA VI Conference proceedings*, pages 1–7. Usenix, OCT 1992.

D. K. Arvind. Detection of concurrency-related errors in joyce. In *Proceedings of Second Joint International Conference on Vector and Parallel Processing*, volume 634, pages 127–132. Springer-Verlag, 1992.

D. K. Arvind. On the detection of communication-related errors in concurrent programs. *Parallel Computing*, 18:1381–1392, 1992.

D. K. Arvind and J. Knight. Detection of concurrency-related errors in joyce. In *Second Joint International Conference on Vector and Parallel Processing*, pages 127–132. Springer-Verlag, SEP 1992.

D. K. Arvind and C. Smart. Hierarchical parallel discrete event simulation in composite ELSA. In *Proceedings of the 6th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation, Newport Beach, USA.*, pages 147–156. Society for Computer Simulation, JAN 1992.

A. Avron, F. Honsell, I. Mason, and R. A. Pollack. Using typed lambda calculus to implement formal systems on a machine. *Journal of Automated Reasoning*, pages 309–354, 1992.

D. M. Berry, A. J. R. G. Milner, and D. N. Turner. A semantics for ML concurrency primitives. In *Conference Record of the 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 119–129. ACM Press, JAN 1992.

J. C. Bradfield. A proof assistant for symbolic model-checking. In *Proceedings CAV '92*, 1992.

J. C. Bradfield and C. P. Stirling. Local model checking for infinite state spaces. *Theoretical Computer Science*, 96(1):157–174, APR 1992.

G. Bruns. A case study in safety-critical design. In *Proceedings of CAV '92*, page 12, JUL 1992.

CST-99-93 (also published as ECS-LFCS-93-266)

Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms

Davide Sangiorgi

Abstract: We study *mobile systems*, i.e. systems with a dynamically changing communication topology, from a process algebras point of view. Mobility can be introduced in process algebras by allowing names or terms to be transmitted. We distinguish these two approaches as *first-order* and *higher-order*. The major target of the thesis is the comparison between them.

The prototypical calculus in the first-order paradigm is the *π-calculus*. By generalising its sort discipline we derive an ω-order extension called *Higher-Order π-calculus* (HOπ). We show that such an extension does not add expressiveness to the π-calculus: Higher-order processes can be faithfully compiled down to first-order, and respecting the behavioural equivalence we adopted in the calculi. Such an equivalence is based on the notion of bisimulation, a fundamental concept of process algebras. Unfortunately, the standard definition of bisimulation is unsatisfactory in a higher-order calculus because it is over-discriminating. To overcome the problem, we propose *barbed bisimulation*. Its advantage is that it can be defined *uniformly* in different calculi because it only requires that the calculus possesses an *interaction* or *reduction* relation. As a test for barbed bisimulation, we show that in CCS and π-calculus, it allows us to recover the familiar bisimulation-based equivalences. We also give simpler characterisations of the equivalences utilised in HOπ. For this we exploit a special kind of agents called *triggers*, with which it is possible to reason fairly efficiently in a higher-order calculus notwithstanding the complexity of its transitions.

Finally, we use the compilation from HOπ to π-calculus to investigate Milner's encodings of λ-calculus into π-calculus. We present analogous encodings of λ-calculus into HOπ. By comparison with those into π-calculus, these are easier to understand and with a closer correspondence between reduction on λ-terms and on their process counterparts. We show that the two encodings of lazy λ-calculus and the compilation from HOπ to π-calculus commute. Thus we can reason in HOπ and the results hold for π-calculus as well. In this way we are able to derive a direct characterisation of the equivalence upon λ-terms induced by Milner's encoding and by the behavioural equivalence adopted on the process terms.

From the representability result of HOπ in π-calculus we conclude that the first-order paradigm, which enjoys a simpler and more intuitive theory, should be taken as *basic*. Nevertheless, the study of the λ-calculus encodings shows that a higher-order calculus can be very useful for reasoning at a more abstract level.

Rod Burstall, Director; Dr. George Cleland, Assistant Director; and Profs. Mike Fourman, Robin Milner and Gordon Plotkin, Co-Directors. An Advisory Board composed of representatives of the rank and file of the Laboratory, internal and external academics and industrialists helps to provide a strategic direction. The research committee, composed of all academic staff members of LFCS, and the business meeting, composed of all members of the Laboratory, meet regularly and provide further input to the direction and running of the Laboratory.

Individual grant management is the responsibility of investigators, but LFCS provides a strong cultural environment to support this research. We have recently instigated a formal research appraisal mechanism. Each of the themes mentioned below is appraised every two years. This has been found to be most effective in broadening awareness between projects and individuals and provides valuable input to guide our research strategy. The "Lab-Lunch" provides a weekly forum for research communication in LFCS with each member giving a short presentation. This is supplemented by research "clubs" in specific areas, and a more formal seminar series.

LFCS Research Programme

The mainspring of research in LFCS is the study of theories which underlie, or should in future underlie, the analysis and design of computing systems. Many theories are under present study: general semantic theories, theories specifically for concurrent systems, theories for system specification, theories of programming language design, and theories of general logic (to underlie computer-assisted system analysis). In almost all these cases, a significant software tool has been or is being built to mediate the theory to applied computer scientists, including industrial scientists.

Research in LFCS is organised under a number of broad themes, **Logics and Proof**, **Concurrency**, **Language and Semantics**, **Complexity and Algorithms**, and **Formal Development of Programs and Systems**. These are described individually earlier in this Report under Research. It is worth noting that they unite into a large and coherent research effort with considerable cross fertilisation between projects and across themes. This effort has a core of theoretical research and a practical component which explores application and implementation of the theory. The research is unified by a common culture. It is also unified by the predominant use of the same programming language (Standard ML) in software development. Standard ML is in widespread use in Academia and is beginning to diffuse into Industry.

Applied research covers the development of specific theories, the analysis of specific systems and the embodyment of theories in a range of tools. Work on analysing "real" systems helps diffuse theoretical techniques into practice in Academia and Industry. Tools provide machine assistance, for example in analysing concurrent systems or carrying out formal proofs. These systems or their descendants are typically used by software engineers and provide methods which allow rigorous

proof of properties of the resulting system. These tools must support reasoning in a variety of mathematical logics, and almost half of the Laboratory's resource is committed to projects concerned with either building the computational tools for constructing proofs or tackling the mathematical problems in integrating a number of logics into a single framework.

Advanced tools developed in LFCS requires powerful state of the art computing facilities. As well as being supported by both the University and funding agencies, Digital and Hewlett Packard have each provided significant resources.

Much of the systems work of LFCS is exported through its system dissemination programme. Systems are available at cost (or free by FTP) for research or educational use, with commercial use being subject to individual licencing agreement. A number of sub-licencing arrangements are in place.

LFCS research is funded from a variety of sources. SERC continues as our main sponsor, but the proportion of funding from other agencies has increased in recent years. These include UK Dept. of Trade and Industry; EC agencies such as ESPRIT, and The Human Capital and Mobility Scheme; the British Council; and, of course, industry, both in the UK and abroad.

Industrial Interaction

One element of the Laboratory which distinguishes it from merely being a collection of research projects is the formal structures which it uses to interact with industrial research organisations. This takes place in a number of ways:

The **Affiliation Programme** provides, to Industrial R & D groups, a low cost means of keeping in touch with both the work of the Laboratory. The Laboratory's **Course Programme** is now strongly developed with half a dozen course in its portfolio. **Industrial Visitors** are encouraged to spend periods of time in the Laboratory. A number of **Collaborative Projects** are in place in several of the research themes above.

The Laboratory publishes a widely acclaimed report series comprising both research and expository material. Copies of these and other publications can be obtained from George Cleland, the Assistant Director, from whom further general information on the Laboratory may also be obtained.

Contact Information for LFCS:
E-mail:`lfcs@ed.ac.uk`

LFCS Software (and LFCS reports from Jan '94) is available by anonymous FTP:
`ftp.lfcs.ed.ac.uk`

(log on as anonymous – use net address as password, look in directory `export`)

CST-98-93

PAC-Learning Geometrical Figures Paul W Goldberg

Abstract: The thesis studies the following problem: Given a set of geometrical figures (such as planar polygons), each one labelled according to whether or not it resembles some "ideal" figure, find a good approximation to that ideal figure which can be used to classify other figures in the same way.

We work within the PAC learning model introduced by Valiant in 1984. Informally, the concepts under consideration are sets of polygons which resemble each other visually. A leaning algorithm is given collections of members and non-members of a concept, and its task is to infer a criterion for membership which is consistent with the given examples and which can be used as an accurate classifier of further example polygons.

In order to formalise the notion of a concept, we use metrics which measure the extent to which two polygons differ. A concept is assumed to be the set of polygons which are within some distance of some fixed central polygon. In the thesis we work most extensively with the Hausdorff metric.

Using the Hausdorff metric we obtain NP-completeness results for several variants of the learning problem. In particular we show that it is hard to find a single geometrical figure which is close to the positive examples but not to the negative examples. This result holds under various assumptions about the specific geometrical figures under consideration. It also holds for several metrics other than the Hausdorff metric.

Despite the NP-completeness results mentioned above we have found some encouraging positive results. In particular, we have discovered a general technique for *prediction*. (Prediction is a less demanding learning model that PAC learning. The goal is to find a polynomial time.) Using our technique we have obtained polynomial-time algorithms for predicting many of the geometrical concept classes studied in the thesis. These algorithms do not classify geometrical figures by measuring their distance from a single "ideal" geometrical figure. Instead, they identify a collection of concepts whose intersection may be used to classify examples reliably.

It is natural to consider the case in which only positive examples are available. In the thesis we show that some but not all of the concept classes may be predicted from positive examples alone.

We consider prediction to be a useful goal, since it solves the practical problem of classifying unlabelled examples. However in the final section of the thesis we show a theoretical limitation to the effectiveness of this technique. In particular, assuming the existence of trapdoor functions, not polynomial-time algorithm for prediction exists for polygons in the plane which are assumed to be equivalent under classes of isometrics that include rotations.

CST-97-93 (also published as ECS-LFCS-93-255)
A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification Michael Mendler

Abstract: The application of formal methods to the design of correct computer hardware depends crucially on the use of abstraction mechanisms to partition the synthesis and verification task into tractable pieces. Unfortunately however, behavioural abstractions are genuine mathematical abstractions only up to behavioural *constraints*, i.e. under certain restrictions imposed on the device's environment. Timing constraints on input signals form an important class of such restrictions. Hardware components that behave properly only under such constraints satisfy their abstract specifications only approximately. This is an impediment to the naive approach to formal verification since the question of how to apply a theorem prover when one only knows *approximately* what formula to prove has not as yet been dealt with.

In this thesis we propose, as a solution, to interpret the notion of 'correctness up to constraint' as a modality of intuitionistic predicate logic so as to remove constraints from the specification and to make them part of its proof. This provides for an 'approximate' verification of abstract specifications and yet does not compromise the rigour of the argument since a realizability semantics can be used to extract the constraints. Also, the abstract verification is separated from constraint analysis which in turn may be delayed arbitrarily. In the proposed framework constraint analysis comes down to proof analysis and a computational semantics on proofs may be used to manipulate and simplify constraints.

Supported Research

The table on the following page lists funded research in the department during the period of this report. Giving total figures is a little meaningless as the grants start at different times and have different durations. However the table shows that there is a large number of externally funded research projects, that there is a large range of sponsoring bodies, and that a significant number of research staff are employed through external research funding.

The *breadth* of funded research is large. Projects range across theoretical topics such as categorical approaches to computer science, or the use of constructive mathematics to support formal proof, through to more applied areas such as concurrent database research and super-scalar processor design. Increasingly there is a trend towards the use of theoretically sound ideas in applied work. This "crossover" research is typified by work in for example protocols, safety-critical systems, hardware design and performance measurement.

The department also obtains support from a wide range of *funding bodies*. The majority of funding continues to come through the SERC route, but the proportion from elsewhere has increased in recent years. This diversity of support protects us somewhat from the irregularities in funding within any one agency. This allows us to develop a more stable research environment, enhancing both the scientific culture and employment security of research staff. Basic research is funded through SERC grants, ESPRIT basic research actions and the EC Human Capital and Mobility Programme. More applied research is funded through ESPRIT industrial projects, the DTI, and indeed industry itself.

In the future the policy of diversification will continue. A recently started teaching company programme with Adelard in the safety-critical area is intensifying contact with industry. We are pursuing further such liaisons. Other successes include membership of several European networks of excellence (and coordinating site for one network).

There is also a strong record of collaboration with other departments in the University. These include Artificial Intelligence (Proof Systems Research), Physics (through the Edinburgh Parallel Computing Centre) and Sociology (High Integrity Computing). A major project in the safety-critical systems area has recently started. This involves the Human Communications Research Centre and the Departments of Sociology and Artificial Intelligence, as well as major industrial players such as Shell and British Rail, and small companies such as Praxis and Adelard.

Other notable devlopments this year include a project on architectural simulation and modelling which is unifying and focussing several threads of systems reasearch, and a project in developing mammography screening techniques based upon vision techniques initially developed for analysing astronomical images.

Not mentioned on the table are a number of fellowships awarded to permanent staff to relieve them of teaching duties. Profs. Robin Milner and Gordon Plotkin each hold SERC Senior Fellowships (5 years). Dr Don Sannella holds an SERC Advanced Fellowship (5 years). Dr Mark Jerrum holds a Nuffield Fellowship.

Each of these grants allows us to appoint replacement teaching staff (1 year). This has allowed us to maintain reasonable strength in critical areas through the University's current financial difficulties.

CST-102-93

Task Assignment in Parallel Processor Systems

Sathiamoorthy Manoharan

Abstract: This thesis studies the problem of assigning programs onto parallel processor systems. It develops a generic simulation environment to model parallel systems and uses this environment to assess assignment techniques.

Graphs are used in modelling programs, and based on these program models, a taxonomy for assignment shemes is proposed. Assignment schemes are broadly classified into schemes dealing with dependency graphs and schemes dealing with interaction graphs. Desirable properties for efficient assignments under different program models are discussed.

In contrast to the assignment of an interaction graph, an assignment of a dependency graph, in general, can be proved to be close to the optimal assignment. Moreover, the explicit temporal information made available by dependency graphs helps in establishing better assignment heuristics. The thesis thus focuses on the assignment of dependency graphs.

Most of the published schemes for assigning dependency graphs are work-greedy. Their heuristics is based on satisfying the following rule of thumb: keeping the processors busy will lead to a "good" assignment. These schemes do not let a processor idle if there is a task the processor could execute. New analytical results bounding the performance of work-greedy assignment schemes are derived. It is shown that, when communication costs cannot be ignored, work-greedy assignment schemes may not perform well. An alternative assignment scheme which has a time-complexity lower than those of the work-greedy schemes is proposed.

A generic object-oriented simulation platform is developed in order to conduct experiments on the performance of assignment schemes. The simulation platform, called Genesis, is generic in the sense that it can model the key parameters that describe a parallel system: the architecture, the program, the assignment scheme and the message routing strategy. Genesis uses as its basis a sound architectural representation scheme developed in the thesis.

The thesis reports results from a number of experiments asssessing the performance of assignment schemes using Genesis. The comparison results indicate that the new assignment scheme proposed in this thesis is a promising alternative to the work-greedy assignment scheme. The proposed scheme has a time-complexity less than those of the work-greedy schemes and achieves an average performance better than, or comparable to, those of the work-greedy schemes.

To generate an assignment, some parameters describing the program model will be required. In many cases, accurate estimation of these parameters is hard. It is thought that inaccuracies in the estimation would lead to poor assignments. The thesis investigates this speculation and presents experimental evidence that shows such inaccuracies do not greatly affect the quality of the assignments.

CST-104-93

A Performance Monitoring and Analysis Environment for Distributed Memory MIMD Programs

Kayhan Imre

Abstract: This thesis studies event monitoring techniques that are used for collecting, filtering and visualising event traces from parallel programs. Implementations of two experimental monitoring systems are presented. The first system is a hybrid implementation which uses extra hardware to collect event traces. The second system is a software implementation which was implemented on the Edinburgh Concurrent Supercomputer. These two systems can gather event traces from the parallel programs at a very low cost. The event abstraction mechanism is used for filtering these event traces. The generic and application specific performance metrics are achieved by using event abstraction techniques to replace event patterns with new abstract events which are used for visualising performance related behaviour. The strengths and weaknesses of the event abstraction approach are discussed in the context of performance analysis and visualisation of message passing parallel programs.

Project	Grant Holder(s)	Value (£)	Funding Body	Start	End
Mathematically Proven Safety Systems	Cleland,Walker,Anderson	228,849	DTI/SERC	1-Jan-90	30-Jun-93
Formal Systems Design	Fourman	72,910	DTI/SERC	1-Jan-90	31-Dec-92
Formally Based Systems Design Tools	Fourman	91,366	SERC	1-Jun-90	30-May-93
Declarative Languages and Applied Semantics	Milner, Fourman, Mitchell	391,230	SERC	1-Apr-90	30-Oct-94
Parallel Processing within Intelligent Knowledge Bases	Thanisch, Wallace	89,896	SERC	1-Jun-90	31-May-93
Logical and Semantical Frameworks	Plotkin, Moggi	329,035	SERC Rolling	1-Jul-90	30-Jun-94
Category Theory in Computer Science	Jay	110,068	BP/RSE Fellowship	1-Oct-90	30-Sep-93
Constructive Logic as a Basis for Program Development	Burstall	336,192	SERC Rolling	1-Jan-91	30-Dec-94
Formal Infusion of Concurrency and Communication into Programs	Langauer, Arvind,Fourman	106,693	SERC	4-Jan-91	31-Mar-94
Formal Reasoning Awareness Club	Cleland	33,487	DTI Contract	1-Jul-91	30-Jun-95
Verification of Infinite State Systems	Stirling, Bradfield	85,513	SERC	1-Oct-91	30-Sep-94
Brit-Spanish Collaboration Project	Sannella	1,950	Brit. Council	7-Oct-91	30-Jun-93
Quantitive Analysis of Stochastic Systems	Sinclair, Jerrum	93,072	SERC	1-Jan-91	31-Dec-93
Postdoc Fellowship	Gardner	28,000	SERC	1-Jan-92	31-Dec-93
Postdoc Fellowship	McKinna	28,000	SERC	1-Mar-92	28-Feb-94
Decoupled Architectures for High-Performance Computing	Topham	178,300	EC - ESPRIT	1-May-92	30-Apr-95
Types, Proofs and Programs	Burstall, Plotkin	234,646	EC BRA	1-Sep-92	31-Aug-94
Concurrency and Functions: Evaluations and Reductions	Milner	118,110	EC BRA	1-Sep-92	31-Aug-92
Calculi and Algebras of Concurrency: Extensions, Tools and Applications	Stirling, Milner	56,393	EC BRA	1-Sep-92	31-Aug-94
Typed Lambda Calculus Network	Plotkin	58,333	EC HCM Network	1-Sep-93	31-Aug-96
Programming Language Semantics and Program Logics - Twinning Programme	Plotkin, Moggi	59,500	CEC Science	1-Sep-92	31-Aug-95
British-German Collaboration Project	Fourman	2,725	Brit. Council	1-Oct-92	30-Sep-94
Development of High Integrity Systems	Anderson, MacKenzie	139,118	ESRC/SERC	1-Oct-92	31-Aug-95
Formal Development of Modular Programs from Algebraic Specifications	Sannella	27,752	SERC	1-Oct-92	30-Jun-93
A Comprehensive Algebraic Approach to System Specification and Development	Sannella, Burstall	15,748	CEC BRA	1-Sep-92	31-Aug-95
Randomized Algorithms	Jerrum	15,358	CEC BRA	1-Sep-92	31-Aug-95
Sannella SERC Advanced Fellowship - Computing Support	Sannella	7,434	SERC	1-Jan-93	31-Dec-93
VMS File System Project	Arvind	17,500	Digital Equipment Co.	8-Feb-93	7-Jun-93
Algebraic and Logical Foundations of Formal Software Development	Sannella	116,625	SERC	16-Feb-93	15-Feb-96
Novel Architecture Computing Commitee	Ibbett, Wallace	70,000	SERC	1-Apr-93	31-Mar-94
Structured Software Systems, Specifications and Logical Systems	Sannella (Tarlecki)	6,240	EC-COST fellowship	1-Apr-93	30-Jun-93
Formal Development of Modular Progs in Extended ML	Sannella	142,409	SERC	1-Apr-93	31-Mar-96
Randomness and Computation	Jerrum	16,680	SERC	1-Jun-93	30-Sep-93
Teaching Company Scheme with Adelard	Cleland, Anderson	166,619	SERC/TCS	21-Jun-93	20-Jun-96
Communication in Safety Cases - A Semantic Approach	Anderson, Cleland	586,928	SERC	1-Jul-93	30-Jun-96
Infinite Computation and Non-determinism in Type Theory	Sannella (Hofmann)	57,552	EC HCM Fellowship	1-Aug-93	31-Jul-95
Simulation Experiments in Parallel Systems Design	Ibbett,Heywood,Cole,Pooley,Thanisch,Topham	221,951	SERC	1-Sep-93	31-Aug-96
Computer Based Mammography Screening	Proctor, Thanisch	180,000	SERC	1-Oct-93	30-Sep-94
Nuffield Support Fellowship	Heywood	4,000	Nuffield	1-Oct-93	30-Sep-94
Shared Environments for Distributed ML	Fourman (Le Sergent)	43,048	EC HCM Fellowship	1-Oct-93	30-Sep-94
Interaction Theories and Concurrency	Milner (Esik)	6,320	EC-COST Fellowship	1-Oct-93	31-Dec-93
Applied Formal Methods Feasibility Study	Cleland	40,044	DTI	1-Oct-93	31-Jul-94
MEDICIS - EC Network of Excellence in Foundations of Software Specification	Sannella	16,357	EC HCM Network	1-Nov-93	31-Oct-96
ropean Institute in the Logical Foundations of Computer Science - Infrastructure	Plotkin	100,000	EC HCM Network	1-Dec-93	30-Nov-96
European Institute in the Logical Foundations of Computer Science - Fellowship Programme	Plotkin	417,000	EC HCM Fellowships	1-Jan-94	30-Dec-96
Frameworks for Specifying Logical Systems	Gardner	75,667	BP/RSE Fellowship	1-Jan-94	31-Dec-96
Distributed Shared Environments for Concurrent ML	Fourman, Mitchell, Ibbett	194,495	SERC	1-Feb-94	31-Jan-97

Industrial Collaboration

The Department has a wide range of activities involving industry. Its strength in direct collaboration is shown by the range of projects outlined below. There are however a number of of other mechanisms adopted to faciltate interaction and the diffusion of technology and concepts into industry. These include:

- An industrial seminar series, designed to forge closer ties with technology-related companies. These seminars provide a forum for discussion on current research directions within the Department and an opportunity for delegates to share common problems and concerns.
- The LFCS industrial affiliation scheme.
- A programme of short courses for industry
- Strong representation from industry on policy committees, covering both teaching and research.

Projects

The following companies are formally involved in collaborative research with the department.

Abstract Hardware, Dowty Controls, European Silicon Structures: Collaborative project project on formal system design; SERC/IED funded.

Adelard: Teaching Company Scheme in the concerned with the design of safety-critical industrial control systems.

Advanced Computer Research Institute (Toulouse, France): Research on the development of decoupled architectures in high performance computing.

Algotronix Ltd: Supply of PC/AT board containing seven Configurable Array Logic (CAL) chips (with conditional offer of a further eighteen CAL chips), for the Bit Stream Enhancement (BSE) project to produce configurable communications hardware. Scottish Enterprise funded.

Digital Equiment Co: Collaboration on the formal analysis and design of distributed file systems for future operating systems.

European Computer-industry Research Centre, Munich Research on industrial strength concurrent programming languages.

European Workshop on Industrial Computer Systems : This is the principal European forum for Industrial Safety and Security. We are actively involved in the generation of pre-standards guidelines in the area of formal methods, distributed systems, and programable logic controllers.

CST-103-93 (also published as LFCS-93-277)

Fibrations, Logical Predicates and Indeterminates

Claudio Alberto Hermida

Abstract: Within the framework of categorical logic or categorical type theory, predicate logics and type theories are understood as fibrations with structure. Fibrations, or fibred categories, provide an abstract account of the notions of indexing and substitution. These notions are central to the interpretations of predicate logics and type theories with dependent types or polymorphism. In these systems, predicates/dependent types are indexed by the contexts which declare the types of their free variables, and there is an operation of substitution of terms for free variables.

With this setting, it is natural to give a category-theoretic account of certain logical issues in terms of fibrations. In this thesis we explore logical predicates for simply typed theories, induction principles for inductive data types, and indeterminate elements for fibrations in relation to polymorphic λ-calculi.

The notion of logical predicate is a useful tool in the study of type theories like simply typed λ-calculus. For a categorical account of this concept, we are led to study certain structures of fibred categories. In particular, the kind of structure involved in the interpretation of simply typed λ-calculus, namely cartesian closure, is expressed in terms of adjunctions. Hence we are led to consider adjunctions between fibred categories. We give a characterisation of these adjunctions which allows us to provide structure, given by adjunctions, to a fibred category in terms of appropriate structure on its base and its fibres.

By expressing the abovementioned categorical structure logically, in the internal language of a fibration, we can give an account of logical predicates for a cartesian closed category. By recourse to the internal language, we regard a fibred category as a category of predicates. With the same method, we provide a categorical interpretation of the induction principle for inductive data types, given by initial algebras for endofunctors on a distributive category.

We also consider the problem of adjoining indeterminate elements to fibrations. The category-theoretic concept of indeterminate or generic element captures the notion of parameter. Lambek applied this concept to characterise a functional completeness property of simply typed λ-calculus or, equivalently, of cartesian closed categories. He showed that cartesian categories with indeterminate elements correspond to Kleisli categories for suitable comonads. Here we generalise this result to account for indeterminates for cartesian objects in a 2-category with suitable structure. To specialise this 2-categorical formulation of objects with indeterminates via Kleisli objects to the 2-category $\mathcal{F}ib$ of fibrations over arbitrary bases, we are led to show the existence of Kleisli objects for fibred comonads. These results provide us with the appropriate machinery to study functional completeness for polymorphic λ-calculi by means of fibrations with indeterminates. These are also applied to give a semantics to ML module features: signatures, structures and functors.

CST-100-93 (also published as ECS-LFCS-93-268)

Embedding Hardware Description Languages in Proof Systems

Kees G W Goossens

Abstract: The aim of this thesis is to investigate the integration of hardware description lamguaages (HDLs) and automated proof systems.

Simulation of circuit designs written in an HDL is an important method of testing their correctness. However, due to the combinatorial explosion of possible inputs it is not feasible to verify designs using simulation alone. Formal hardware verification, using a proof system, has tried to address this issue. Whilst some medium-sized designs have been (partially) verified, industrial take-up of formal methods has been slow. This is partly due to the use of specialised, non-standard notations employed in various formalisms.

By embedding a hardware description language in a proof system we hope to clarify the semantics of the particular HDL, and present a more standard interface to formal methodologies. We have given a new static structural operational semantics for a subset of the ELLA hardware language. The formal dynamic semantics of this subset is based on an existing informal model.

We embedded the semantics of this HDL in the LAMBDA higher-order logic proof system. The embedding allows meta-theoretical results to be proved about this and other semantics. It has been proved that the semantics computes the least fixed point solution of the circuit description. Another semantics which computes a more defined output has also been embedded, and the relationship between both semantics has been proved formally.

A number of paradigms such as operational semantics based formal symbolic simulation, formal interactive (top-down and bottom-up) synthesis, formal hardware generators, proved correct transformations and traditional hardware verification are presented as small case studies. However, scaling up of the examples turned out to be difficult and verification tended to be slow.

Harlequin: Collaboration on the use of type theory and algebraic specification in the design and analysis of compiler sub-systems and advanced application software.

Motorola: Collaborative project, with Motorola (East Kilbride) on generating an architecture of parallel DSP chips to realise a range of applications. The donation includes several DSP96002 processors and supporting software.

Philips: Research on the use of constructive type theory in development of communications protocols.

Royal Observatory Edinburgh, Wolfson Image Analysis Unit: Collaboration on the use of computer based vision techniques for mamographic screening.

Trustee Savings Bank and Ordnance Survey: Collaboration on database performance and "usability", and database integration of geographical information systems respectively. SERC/DTI funded.

Shell Expro, Praxis, British Rail, AEA Technologies, Lucas Automotive, Adelard, Collaboration on the role of formal and informal semantics, in the design, construction and assessment of safety-related systems.

PhD Support

Siemens (Munich) funds two postgraduate students

Hewlett Packard funds one overseas postgraduate student

The following companies support CASE students:

Digital (Scotland)

Harlequin

Inmos

DRA Malvern

Research Degrees

Students studying for the degree of PhD undertake full-time, supervised research for a minimum period of 36 months and 12 months in the case of the research MSc.[3] This research will normally be in an area of an existing specialism within the Department (see section on *Research Interests* for further information) but interesting proposals in other areas receive consideration.

[3]There is also an M.Phil. which demands full-time study for a minimum period of 24 months

The PhD "programme" includes a first year of study in which students are registered as *Supervised Postgraduate Students* prior to registration as PhD candidates. During this first year, students are expected to participate in an appropriate study programme. For students registered for research in the computational theory area, a specific course on advanced topics is provided. Students intending to pursue research into computer systems or other non-theoretical topics may be directed to particular advanced courses and are expected to attend postgraduate study seminars. In order to register as a PhD candidate, students must submit for approval a research proposal which is discussed with a small panel of staff. The degree itself is examined on the basis of a thesis and oral examination[4]. Closing date for PhD applications is 31 March 1994.

Abstracts of PhD Theses

The following pages reproduce the abstracts for the theses published by the department in this academic year. The variety of topics covered should give the reader some idea of the range and diversity of PhD work undertaken at Edinburgh.

[4]Further PhD application-related information may be obtained from Miss E Kerse at the Departmental address, or tel. 031 650 5156, fax. 031 667 7209, e-mail: eak@uk.ac.ed.dcs

CST-101-93 (also published as ECS-LFCS-93-271)

Timed Processes: Models, Axioms and Decidability

Liang Chen

Abstract: This thesis presents and studies a timed computational model of parallelism, a Timed Calculus of Communicating Systems or Timed CCS for short. Timed CCS is an extension of Milner's CCS with time. We allow time to be discrete, such as the natural numbers, or dense, such as the non-negative rationals or the non-negative reals. We make no assumption of the Maximal Progress Principle, but the calculus is consistent with the principle. Time variables in the language allow us to express a notion of time dependency and the language is more expressive than those without time variables or infinite summation. We extend the well known notion of bisimulation to timed processes and study the abstract sensation of timed processes. We show that strong equivalence (the largest strong bisimulation) is decidable for finite processes, i.e. processes without recursion. The decidability is independent of the choice of time domain. We also present a simple proof system for strong equivalence and the proof system is again independent of the choice of time domain. We show that the proof system is sound and complete for finite processes over dense time domains, but only complete for a restricted language over discrete time domains. We discuss how to modify the definition of time expressions to get the restricted language. We also study behavioural abstraction in timed processes.

The thesis also presents and studies a general model, Timed Synchronisation Trees, for timed calculi. Timed synchronisation trees are extensions of synchronisation trees with time. All constructions on timed synchronisation trees are continuous with respect to a natural complete partial order. We can interpret a wide range of real-time process algebras in timed synchronisation trees. As an example, we give a denotational semantics for Timed CCS based on timed synchronisation trees. We show that the denotational and operational semantics of Timed CCS coincide.

CCS is a symbolic calculus in the sense that it treats solely the observation of events of a system. The relative time, location and duration of events are abstracted away from the consideration. If we postulate that every action has a non-zero constant duration, we can observe the usual notions of causality, concurrency and conflict relations of events of a system. By interpreting CCS in Timed CCS based on a postulation that for any two events which are causally related there is at least a non-zero constant delay between them, we get a timed semantics for CCS. The timed semantics of CCS is a partial order or true concurrency semantics. As a consequence, we develop a partial order or true concurrency semantics based on an interleaving approach.