designed by visual resources, the university of edinburgh

Departmental Report 1993-94

DEPARTMENT OF COMPUTER SCIENCE

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 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.

Computer Science Departmental Report

1993 - 1994

Contents

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.
Andrew Wilson *K. Mitchell, G.D. Plotkin*	Programming language semantics.
Kann-Jang Yang *R.J. PooleyFormalisation*	of the software engineering process.
Thomas Zurek *P. Thanisch*	Optimisation in parallel databases.

Ph.D. Graduates

T. Altenkirch: *Constructions, inductive types and strong normalization*

S. Christensen: *Decidability and decompostion in process algebras*

J. Hillston: *A compositional approach to performance modelling*

K. Imre: *A performance monitoring and analysis environment for distributed memory MIMD programs*

P. Martin: *Adding safe and effective load balancing to multicomputers*

M. Norman: *Static allocation of computation to processors in multicomputers*

J. Phillips: *The effectiveness of dynamic process allocation strategies within an occam environment*

A. Welsh: *Effective software support for chemical research*

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*	A motion-sensitive approach to combating inherent latencies in virtual reality systems.
Sandy Robertson *R.N. Ibbett, N.P. Topham*	An object-oriented design environment to simulate and animate microprocessor experiments.
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, James McKinna*	Co-development of imperative programs and their correctness proofs in a type-theoretic environment
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*	Category theory, domain theory, logic, type theory.
Colin Smart *D.K. Arvind*	Parallel simulation; distributed algorithms.
Jitka Stribrna *M.R. Jerrum*	Topics in logic and computational complexity.
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.

The Department

It is pleasing to be able to report that undergraduate teaching in Informatics at Edinburgh, in the Departments of Artificial Intelligence and Computer Science, was assessed as "excellent" in the recent Teaching Quality Assessment excercise carried out by the Scottish Higher Education Funding Coucil (SHEFC). This confirms our position amongst the leading UK Computer Science Departments, not only in research, but also in teaching.

The importance of computing as a discipline within higher education in Scotland has been recognised by SHEFC, which has disaggregated the funding for teaching in computing and mathematics. Furthermore, computing is now to be funded at the same level as science, in recognition of the fact that computing needs to be supported as a laboratory based subject.

One of our concerns in teaching in recent years has been the relatively poor state of our first-year accommodation on Level 3 of the Appleton Tower. This problem has now been solved by a reorganisation and complete refurbishment of the space on Level 3, thus providing us with a well laid out and very attractive area containing the main laboratory along with consultation and work rooms.

A new innovation in teaching this year is the introduction of a B.Eng. degree in Software Engineering. The first two years of this degree will be identical with the B.Sc./B.Eng. in Computer Science. In the third and fourth years there will be more constraints on the choice of existing modules, and a new third year module on Software Systems Design will be compulsory.

The departure of Dr Alistair Sinclair earlier this year to a post at UC Berkeley left us with a gap in the area of computational complexity. We are fortunate to have found an excellent replacement, Dr Vivek Gore, who is moving here from a post at UCLA.

We are sad to be losing Professor Robin Milner from the Department, following his appointment to a chair at Cambridge. Robin has made outstanding contributions both to the discipline and to the Department. We wish him and his family well in their new surroundings. Recognising the importance of the subject he has helped to create, the University has established a permanent Chair in Theoretical Computer Science which we plan to fill during the course of the next academic year.

Professor Roland N. Ibbett, Head of Department
August 1994

Congratulations are due to Roland Ibbett on being elected a Fellow of the Royal Society of Edinburgh, and on his appointment as a University Vice-Principal from October 1994. The loss of half of Professor Ibbett's time as a consequence of this appointment will being made up by the part-time appointment of Roopa Rangaswami, whom we are pleased to welcome.

The administrative structure of the Department has also been reorganised in the light of these changes, and general responsibility for four broad areas — infrastructure, personnel, teaching and research — devolved to *area convenors*. These convenors (including the Head and Deputy Head of Department) meet regularly to coordinate departmental policy and its implementation.

Dr Gordon Brebner, Deputy Head of Department
August 1994

Christopher Lüth *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*	The use of skeletons in algorithmic models of parallel computation.
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.
Alexander Mifsud *S.O. Anderson, R. Milner*	Models of concurrency.
Alvaro Moreira *K. Mitchell, D.T. Sannella*	Theoretical and practical aspects of proof systems for reasoning about type systems and functional programs.
Matthew Morley *R. Milner, G. Cleland*	Understanding "safety" in safety critical software and systems through industrially relevant case studies.
Andrew Morris *S.O. Anderson, G.L. Cleland*	Applications of model checking to safety critical systems.
Isabel Rojas Mujica *P. Thanisch, R. Pooley*	Performance oriented parallel program design and specification.
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, Iain Lindsay (Elec Eng)*	Self-Timed Field Programmable Gate Array (FPGA) Systems.
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.

Rycharde Hawkes *R.A. McKenzie*	Software architectures for modeling and supporing distributed artificial environments.
Edward Heal *G. Brebner*	Formal verification of computer communication protocols.
Timothy Heap *R.M. Burstall*	Computer aided formal reasoning and mathematical foundations.
Roberto Hexsel *N.P. Topham, S.O. Anderson*	Architecture of logically shared, physically distributed memories for scalable multiprocessors; IEEE's Scalable Coherent Interface.
Martin Hofmann *D.T. Sannella, G. D Plotkin*	Infinite computations and nondeterminism in type theory.
Anna Hondroudakis *R.N. Procter, P. Thanisch*	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.
Kai Jakobs *R.N. ProcterUser*	requirements in data communications services
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, R.N. Ibbett*	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.
Budi Ling *M. Fourman*	Category theory, logic, and computation.
John Longley *M.P. Fourman, G.D. Plotkin*	Applications of topos theory to programming language semantics.

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 contains two major research units, the Laboratory for the Foundation of Computer Science (LFCS) and the Computer Systems Group (CSG).

The LFCS is a federation 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 CSG is the focus for research into computer systems within the Department. Its areas of interest concern study and application of techniques which are relevant to actual computer systems. At the present these include performance analysis, novel architectures and compiler techniques for these, database technology, human factors in computing, models of parallel computation, communications and networking, graphics and visualisation, and object oriented techniques.

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.

Stephen Cusak *R. N. Ibbett, N. P. Topham*	Evaluation of multiprocessor interconnection networks.
Ewen Denney *G. D. Plotkin*	A general theory of program refinement.
Luis Dominguez *D. T. Sannella, S. O. Anderson*	Foundations for formal 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.
Marcio Fernandes *R.N. Ibbett, N.P. Topham*	Parallel computing algorithms and architectures.
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.
Vashti Galpin *J.C. Bradfield*	Comparative concurrency semantics
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, T. Heywood*	Practical and theoretical aspects of the shared memory model for parallel computing.
Mark Hartswood *R.N. Procter*	Human factors in computer-aided mammography
Masahito Hasegawa *R.M. Burstall*	Type theory and category theory,functional programming languages

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

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

Dr X Yuan: *(China Computer Systems Engineering Research Institute)*
Parallel real-time systems.

Postgraduate Students

Student/Supervisor	Project
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.
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.
Alan Crawford *N.P. Topham, R.N. Ibbett*	Design of high performance decoupled architectures; compilation techniques for decoupled architectures.
David Crooke *R. Candlin, T.H. Heywood*	Methods for architecture-independent parallel software development.

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.

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.

Visiting Scholars

Mr P. Bailey: *(Australian National University)*
Algorithmic Skeletons in paraML.

Ms A. B. Compagnoni: *(Nijmegen University)*
Typed λ-calculi. Higher-order subtyping. Intersection types.

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.

Dr Kostadin D. Kratchanov: *(Technical University of Plovdiv, Bulgaria)*
Rule-Based Systems, Programming Paradigms, Fuzzy Algorithms and Systems

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

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.

Prof A. Tarlecki: *(Warsaw University and 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.

Mrs L.M. McGill (lmm)	CS Administration

Technical Staff

Mr J.C. Dow (jcd)	Laboratory Superintendant
Mr A.M. Duncan (amd)	General workshop
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

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

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.

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): General system development; Emacs editor support.

Ms C. Dow (carol): Secretarial Support, Macs, Accreditation, faults, support.

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

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

Mr D.J. Rogers (ddr): Computer Systems group co-ordinator; CS1 teaching support; VLSI support; Faculty representative for ECAD, Eurochip and EASE

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

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

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

Ms R. Soutar (rs): p/t Computing Officer
World Wide Web; administrative databases; oracle.

Mr R.W. Thonnes (rwt): Staff systems management; 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
Ms M. Davis (mda)	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 Mairi McLennan	MsC admissions

Dr J.H. McKinna (jhm): Interactive theorem proving. Typed λ-calculi, especially as applied to the formalisation of mathematics, in particular proof theory, operational semantics of programming languages, specification and verification of programs, symbolic algebra. Category theory, especially categorical logic, denotational semantics of programming languages, and foundations of mathematics.

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

Dr D. Matthews (dcjm): finishes 30/9/94 lmm (tlc)ok 15/8/94 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.

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

Dr J. L. Underwood (jlu): Program specification and development; logics for reasoning about programs; computational content of proofs.

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, WWW.

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.

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 Performance Group and other special interest groups of the Computer Systems Group, the Complexity Group and the various clubs within the Laboratory for the Foundations of Computer Science (and while CSG and LFCS themselves organise coherent sub-sets 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.

(Senior) Research Workers/Postdoctoral Fellows

Dr S.H. Ar (sigal): Correctness and reliability of computations, within an algorithms and complexity context.

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

Dr G. Chochia (gac): Modelling of parallel architectures for shared memory simulation.

Ms A. B. Compagnoni (abc): Typed λ-calculi; higher-order subtyping; intersection types; computer assisted program verification and development.

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

Dr P.A. Gardner (pag): Type Theory and Models of Computation: logical frameworks, action calculi and reduction strategies.

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.

P. Heywood (peh): Simulation support environments and tools, object-oriented databases.

Dr Jane Hillston (jeh): Stochastic process algebras; Performance modeling; Markov processes.

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

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

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

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 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): Object-oriented database systems, image analysis, simulation and parallel computing — performance modelling.

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 systems performance evaluation; flow and congestion control in 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 f plagiarism in practical assignments; data structures — sorting and searching algorithms.

Professor Christian Lengauer(lengauer@de.uni-passau.fmi): Infusion of parallelism into specifications and programs, parallelizing compilation, regular and systolic array design, formal semanctics 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

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 is in progress on the human factors of computer-aided digital mammography. This project is a collaboration with radiologists in the U.K. breast screening programme. It involves the design and evaluation of specific user interfaces, and the detailed study of work practices in screening mammography.
- 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 in integrating it effectively with the more traditional forms of business expertise such as that held by managerial and administrative staff.

Rob Procter

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.

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

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 Formalism 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; 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.

Models and calculi for concurrent computation; modal and temporal logics with fixed points and their applications to verification and description of program properties.

Dr D.T. Sannella (dts): *(SERC Advanced Research Fellow; Co-Director, LFCS)*
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): *(Coordinator for Infrastructure, Enterprise coordinator, 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; behavioural synthesis; hardware description languages; 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; configurable cellular arrays.

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

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.

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) allows designers to create architectural designs at different levels of abstraction, to explore designs through a graphical interface and to invoke the appropriate simulator at each level. The output of a simulation run can be used to animate the graphical display of the design, allowing the user to visualise what is happening inside the architecture as programs are run on it. 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 HASE uses the object oriented simulation language Sim++ and an object oriented database, ObjectStore, to store both the object libraries and the designs created from them.

Roland Ibbett, Nigel Topham, Todd Heywood, Murray Cole, Pat Heywood Rob Pooley, Peter Tanisch, George Chochia

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 system is also being simulated using HASE.

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

Evaluation of Multiprocessor Interconnection Networks

Message passing networks are now an established part of multiprocessor systems design, with machines incorporating such networks being produced by a variety of vendors. The networks used vary dramatically both in topology and protocol, and in the design of the interconnect switching components from which they are constructed.

In order to assess the relative value of these networks it is necessary to ascertain the variation in system performance with changes in the network. It is inappropriate to run a simulation of a complete parallel machine, so in this project HASE is used set up a testbed containing simple processor models which can generate network activity corresponding to that found in standard benchmarks used in the evaluation of real parallel systems.

Full models of the various networks under test can then be instantiated in the testbed and simulations run. This arrangement will allow both the impact of different networks on system performance and the utilisation of these networks by different algorithms to be assessed.

The knowledge gained will be used in combination with previous experience gained in developing the Xbar network IC to determine the practicality of providing improvements in the network to enhance system performance.

Roland Ibbett. Alex Wight, Douglas Rogers

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.

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

People

The department 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): *(FRSE; 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)*

[6] Subsequent entries provide usernames only

Price £8.50

CST-109-94 A Statistical Investigation of the Factors Influencing the Performance of Parallel Programs, with Application to a Study of Process Migration Strategies
Joseph Phillips
Price £9.00

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 tree-based hidden surface removal, bidirectional ray tracing and virtual reality. Work on BSP trees concentrates on complexity issues of constructing good trees and in restructuring trees for dynamic scenes. Strategies for parallelisation on different architectures are investigated. Close collaboration is maintained with researchers in the Edinburgh Parallel Computing Centre on graphics projects. Bidirectional ray tracing is one such project. Current Virtual Reality (VR) research interests include the modeling and simulation of artificial environments with particular emphasis on the distribution of the simulation over a network of machines. Other VR research issues include an investigation into reducing the latencies of systems based upon the relative motion of objects and the user's head. Work is being conducted in conjunction with the Virtual Environment Laboratory in the Department of Psychology which is looking at issues regarding the construction of artificial environments and how people interact within them. Resources include immersive technology (e.g. Head Mounted Displays) and non-immersive technology (e.g. a simple driving simulator).

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 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

ECS-LFCS-94-299
On the Bisimulation Proof Method
Davide Sangiorgi

ECS-LFCS-94-300
The Definition of Extended ML
Stefan Kahrs, Donald Sannella and Andrzej Tarlecki

ECS-LFCS-94-301
An Old Sub-Quadratic Algorithm for Finding Extremal Sets
Paul Pritchard

ECS-LFCS-94-302
Decidability of model checking for infinite-state concurrent systems
Javier Esparza

ECS-LFCS-94-305
Dynamic load balancing for systems with large and fast variation of load
Thierry Le Sargent

ECS-LFCS-94-306
Dynamic change of protocols for strict coherency in distributed shared memory
Thierry Le Sargent

Theses

CST-103-93	Fibrations, Logical Predicates and Indeterminates Claudio Alberto Hermida Price – £8.00
CST-104-93	A Performance Monitoring and Analysis Environment for Distributed Memory MIMD Programs Kayhan Imre Price – £8.50
CST-105-93	Decidability and Decomposition in Process Algebras Søren Christensen Price £8.50
CST-106-93	Constructions, Inductive Types and Strong Normalization Thorsten Altenkirch Price £8.00
CST-107-94	A Compositional Approach to Performance Modelling Jane Hillston Price £8.50
CST-108-94	Adding Safe and Effective Load Balancing to Multicomputers Paul Martin

ECS-LFCS-94-287
Applying Process Refinement to a Safety-Relevant System
Glenn Bruns

ECS-LFCS-94-288
A polynomial-time algorithm for deciding bisimulation equivalence of normed Basic Parallel Processes
Yoram Hirshfeld, Mark Jerrum and Faron Moller

ECS-LFCS-94-289
Why tricategories?
A J Power

ECS-LFCS-94-290
A very simple algoruthm for estimating the number of k-colourings of a low-degree graph
Mark Jerrum

ECS-LFCS-94-291
Congruences in Commutative Semigroups
Yoram Hirshfeld

ECS-LFCS-94-292
Improved approximation algorithms for MAX k-CUT and MAX BISECTION
Alan Frieze and Mark Jerrum

ECS-LFCS-94-293
On modal mu-calculus and Büchi tree automata
Roope Kaivola

ECS-LFCS-94-294
Deciding equivalences in simple Process Algebras
Yoram Hirshfeld

ECS-LFCS-94-295
Trapping Mutual Exclusion in the Box Calculus
Javier Esparza and Glenn Bruns

ECS-LFCS-94-296
The computational complexity of counting
Mark Jerrum

ECS-LFCS-94-297
A fully abstract semantics for causality in the π-calculus
Michele Boreale and Davide Sangiorgi

ECS-LFCS-94-298
High undecidability of weak bisimilarity for Petri nets
Petr Jančar

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 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

ECS-LFCS-93-275
Multiple Inheritance via Intersection Types
Adriana B Compagnoni and Benjamin C Pierce

ECS-LFCS-93-276
Undecidable Equivalences for Basic Parallel Processes
Hans Hüttel

ECS-LFCS-93-277 (also published as CST-103-93)
Fibrations, Logical Predicates and Indeterminates
Claudio Alberto Hermida
PhD Thesis – Price £8.00

LFCS-93-278 (also published as CST-105-93)
Decidability and Decomposition in Process Algebras
Søren Christensen
PhD thesis – Price £8.50

LFCS-93-279 (also published as CST-106-93)
Constructions, Inductive Types and Strong Normalization
Thorsten Altenkirch
PhD Thesis – Price £8.00

ECS-LFCS-94-280
Higher-Order Subtyping
Martin Steffen and Benjamin Pierce

ECS-LFCS-94-281
Subtyping in $F\,\tilde{\wedge}$ is decidable
Adriana B Compagnoni

ECS-LFCS-94-282
Locality and Non-interleaving Semantics in calculi for mobile processes
Davide Sangiorgi

ECS-LFCS-94-283
Interfaces and Extended ML
Stefan Kahrs, Donald Sannella, Andrzej Tarlecki

ECS-LFCS-94-284
First-Class Polymorphisms for ML
Stefan Kahrs

ECS-LFCS-94-285
The Mobility Workbench. A tool for the π-Calculus
Björn Victor and Faron Moller

ECS-LFCS-94-286
A polynomial algorithm for deciding bisimilarity of normed context-free processes
Yoram Hirshfeld, Mark Jerrum and Faron Moller

[147] P. Yeung and D. J. Rees. Flexibility damping: A strategy for high-level synthesis. In *Proceedings of Synthesis and Simulation Meeting and International Interchange, Nara, Japan*, page 10, OCT 1993.

Reports & Theses 1993-94[5]

DEPARTMENTAL REPORTS

CSR-29-93
List Homomorphic Parallel Algorithms for Bracket Matching
Murray Cole

CSR-30-94
The Performance of SCI Memory Hierarchies
Roberto A Hexsel and Nigel P Topham

COMPUTER SYSTEMS GROUP REPORTS

CSG-2-94
The Formal Specification in Z of Task Migration on the Testbed Multicomputer
Paul Martin

CSG-3-94
The Performance Profiling of a Load Balancing Multicomputer
Paul Martin

CSG-4-94
Comparison of three SPN Packages GreatSPN1.6, DSPNexpress1.2, SPNP3.0
Isabel Rojas Mujica

LFCS Reports

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

ECS-LFCS-93-274
On the decidability of model checking for several μ-calculi and Petri Nets
Javier Esparza

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

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, Stefan Kahrs

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

Dependable Systems

The intellectual core of this area of work is the application of logical and mathematical theories of discrete dynamic systems to a range of 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 the project "Communication in Safety Cases" 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 exploring 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 has developed, new activities have arisen and our intersts have broadened to encompass aspects of dependability of systems. Through a Teaching Company project we are working closely with safety consultants Adelard to develop industrial strength 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 methods to safety systems and on the structure and organisation of high-integrity computing. This year we commence work withing the EU funded OLOS network of excellence on system reliability.

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

Stuart Anderson, Glenn Bruns, George Cleland

[134] A. K. Simpson. A characterisation of the least-fixed-point operator by dinaturality. *Theoretical Computer Science*, 118:301–314, SEP 1993.

[135] A. J. Sinclair. *Algorithms for Random Generation and Counting : a Markov Chain Approach.* Progress in Theoretical Computer Science. Birkhauser Boston, 1993.

[136] A. J. Sinclair and M. R. Jerrum. Polynomial-time approximation algorithms for the ising model. *SIAM Journal on Computing*, 22(4):1087–1116, OCT 1993.

[137] A. J. Sinclair, C. Kenyon, and D. Randall. Matchings in lattice graphs. In *Proc. 25th Symposium on Theory of Computing*, page 9. Association for Computing Machinery, MAY 1993.

[138] A. J. Sinclair, C. Kenyon, and D. Randall. Matchings in lattice graphs. Technical report, International Computer Science Institute, MAR 1993.

[139] A. J. Sinclair, M. Luby, and D. Zuckerman. Optimal speed-up of Las Vegas algorithms. *Information Processing Letters*, 47(4):8, SEP 1993.

[140] A. J. Sinclair, M. Luby, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. In *Proc. 2nd Israel Symposium on Theory of Computing and Systems*, page 6. IEEE, JUN 1993.

[141] A. J. Sinclair, M. Luby, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. Technical Report 10, International Computer Science Institute, MAR 1993.

[142] A. J. Sinclair and D. Randall. Testable algorithms for self-avoiding walks. Technical report, International Computer Institute, AUG 1993.

[143] C. P. Stirling. Modal and temporal logic for processes. Technical report, Aarhus University, 1993.

[144] R. D. Tennent. Correctness of data representations in Algol-like languages. LFCS Report 267, University of Edinburgh, 1993.

[145] N. P. Topham, A. Rawsthorne, and P. Bird. The effectiveness of decoupling. In *Proceedings 7th International Conference on Supercomputing'93*, page 10. ACM, JUL 1993.

[146] A. J. Welsh, R. J. Pooley, A. Welch, G. Mitchell, and D. Welch. Hydride ligand location in complexes of the copper triad and other systems. In *The Chemistry of the Copper and Zinc Triads*, pages 235–239. The Royal Society of Chemistry, AUG 1993.

the HCI'93 Conference, volume People and Computers VIII, pages 383–394. Cambridge University Press, SEP 1993.

[121] R. N. Procter and R. Williams. Beyond design: Social learning and computer-supported cooperative work: Some lessons from innovation studies. In *The Design of Computer-Supported Work and Groupware Systems*, pages 126–144. Elsevier Science, JUN 1993.

[122] D. Pym. Errata and remarks. LFCS Report 265, University of Edinburgh, 1993.

[123] S. Robertson and R. N. Ibbett. HASE: A hierarchical architecture design and simulation environment for computer architects. In *UKSS '93*, page 5. UK Simulation Society, SEP 1993.

[124] D. J. Rogers. An object orientated experimental language for continuous time simulation using C++. In *UKSS'93 First Conference of the UK Simulation Society*, pages 49–53. United Kingdom Simulation Society, SEP 1993.

[125] G. D. M. Ross. Managing X in a large distributed environment. *The X Resource*, 7, 1993.

[126] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.

[127] D. Sangiorgi. A theory of bisimulation for the Pi-calculus. LFCS Report 270, University of Edinburgh, 1993.

[128] D. Sangiorgi. A theory of bisimulation for the Pi-calculus. In *Proceedings of CONCUR'93*, volume 715, pages 127–142. Springer Verlag, 1993.

[129] D. Sangiorgi, M. Gaudel, and J. Jouannaud. From Pi-calculus to higher order Pic-calculus and back. In *TAPSOFT'93*, volume 668, pages 151–166. Springer Verlag, 1993.

[130] D. Sangiorgi and J. Parrow. Algebraic theories for name-passing calculi. In *LICS'93*, 1993.

[131] D. Sangiorgi and J. Parrow. Algebraic theories for name-passing calculi. Lfcs report, University of Edinburgh, 1993.

[132] D. T. Sannella. A survey of formal software development methods. In *Software Engineering: A European Perspective (A. McGettrick and R. Thayer eds.)*, chapter 6, pages 281–297. IEEE Computer Society Press, 1993.

[133] D. T. Sannella and A. Tarlecki. Algebraic specifications and formal methods for program development: what are the real problems? In *Current trends in theoretical computer science (G.Rozenberg and A.Salomaa eds)*, pages 115–120. World Scientific Publishing Company, 1993.

Functional Design and Performance Predicition

This project is developing and using Performance Evaluation Process Algebra (PEPA): a stochastic process algebra for the design of distributed systems. Used as a design tool, a stochastic process algebra encourages the early consideration of the timing characteristics of computer systems. PEPA has been used to model local area networks and multi-server/multi-queue systems, representing performance information and functional information in a single model.

The advantages of such an approach include the natural compositionality of process algebra models and the availability of several novel approaches to model simplification and state space reduction which are based on equivalence relations developed for the PEPA language. The novel combination of performance information and functional information facilitates the study of the performance implications of the functional properties of a system: examples of these include computing the mean time until deadlock or computing the optimal time-out of a service.

Reasoning about PEPA models proceeds by considering the derivation graph obtained from the model using the underlying operational semantics of the PEPA language. The derivation graph is systematically reduced to a form where it can be treated as the state transition diagram of the underlying stochastic (in fact, Markovian) process. From this can be obtained the infinitesimal generator matrix of the Markov process. A steady state probability distribution for the system can then be obtained, if it exists. Alternatively, transient probability distributions may be calculated.

Models can be analysed using the PEPA workbench: a set of simple tools to allow a skilled user of the PEPA language to delegate to machine assistance some of the routine tasks in checking descriptions and performing calculations of transition graphs and rewards.

Future work will consider extensions of traditional model-checking approaches for labelled transition systems to the multi-transition systems used to define stochastic process algebras.

Stephen Gilmore and Jane Hillston.

Formal 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.

D. K. Arvind, Benoit Caillaud, 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.

[108] G. D. Plotkin and M. Abadi. A logical view of composition. *Theoretical Computer Science*, 114(1):3–30, 1993.

[109] G. D. Plotkin, M. Abadi, M. Burrows, and B. Lampson. A calculus for access control in distributed systems. *TOPLAS*, 15(4):706–734, 1993.

[110] G. D. Plotkin, R. Harper, and F. Honsell. A framework for defining logics. *Journal of the Association of Computing Machinery*, 40(1):143–184, 1993.

[111] G. D. Plotkin and G. Huet. *Editor of Logical Environments*. Cambridge University Press, 1993.

[112] G. D. Plotkin and G. Kahn. Concrete domains. *Theoretical Computer Science*, 121:187–278, 1993.

[113] G. D. Plotkin and J. Reynolds. On functors expressible in the polymorphic typed Lambda calculus. *Information and Computation*, 105(1):1–29, JUL 1993.

[114] R. A. Pollack. Closure under Alpha-conversion. In *Informal proceedings of the Nijmegen Workshop on Tyoes for Proofs and Programs, May '93*, pages 329–346, SEP 1993.

[115] R. J. Pooley. A graphical tool for combined simulation and functional modelling. In *Proceedings of the First Conference of the United Kingdom Simulation Society, UKSS '93*, page 5. United Kingdom Simulation Society, SEP 1993.

[116] R. J. Pooley and J. Hillston. *Editor of Computer Performance Evaluation: Modelling Techniques and Tools*, volume 10 of *EDITS*. Edinburgh University Press, AUG 1993.

[117] R. J. Pooley and Richard Zobel. *Editor of Proceedings of the First Conference of the United Kingdom Simulation Society, UKSS '93*. United Kingdom Simulation Society, SEP 1993.

[118] R. N. Procter, A.McKinlay, O.Masting, and R.Woodburn. *Coordination Issues in Tools for CSCW*, volume 4 of *Computer Supported Cooperative Work*. Springer-Verlag, AUG 1993.

[119] R. N. Procter and A. McKinlay. Computer-mediated communications and computer-supported cooperative work. In *Report of the Highland Euroform Conference on Transnational Distance Learning*. Highlands and Islands Enterprise, SEP 1993.

[120] R. N. Procter, A. McKinlay, O. Masting, R. Woodburn, and J. Arnott. A study of turn-taking in a computer-supported group task. In *Proceedings of*

[92] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. LFCS Report 262, University of Edinburgh, 1993.

[93] S. Pelagatti. *A Methodology for the Development and the Support of Massively Parallel Architectures.* PhD thesis, Dipartimento di Informatica Universita di Pisa, 1993.

[94] B. C. Pierce. Intersection types and bounded polymorphism. In *Typed Lambda Calculi ad Applications*, MAR 1993.

[95] B. C. Pierce. Object-oriented programming in typed Lambda-calculus: Exercises and solutions, APR 1993.

[96] B. C. Pierce. Programming in the Pi-calculus: An experiment in programming language design, JUN 1993.

[97] B. C. Pierce, D. Remy, and D. N. Turner. A typed higher-order programming language based on the Pi-calculus. In *Invited lecture at the Workshop on Type Theory and its Application to Computer Systems, Kyoto University*, JUL 1993.

[98] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. In *IEEE Symposium on Logic in Computer Science*, pages 376–385, JUN 1993.

[99] B. C. Pierce and D. N. Turner. Object-oriented programming without recursive types. In *Principles of Programming Languages.* Association for Computing Machinery, JAN 1993.

[100] B. C. Pierce and D. N. Turner. Statically typed friendly functions via partially abstract types. LFCS Report 256, University of Edinburgh, 1993.

[101] G. D. Plotkin. Editor. *Information and Computation*, 1993.

[102] G. D. Plotkin. Editor. *Mathematical Structures in Computer Science*, 1993.

[103] G. D. Plotkin. Editor. *Monograph Series in Theoretical Computer Science*, 1993.

[104] G. D. Plotkin. Editor. *Theoretical Computer Science*, 1993.

[105] G. D. Plotkin. Set-theoretical and other elementary models of the lambda-calculus. *Theoretical Computer Science*, 121:351–410, 1993.

[106] G. D. Plotkin. Type theory and recursion, 1993.

[107] G. D. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Typed Lambda Calculi and Applications*, number 1 in 664, pages 361–375. Springer-Verlag, 1993.

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

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.

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

Semantical Frameworks

The aim of the Semantical Frameworks project is to investigate the mathematical foundations of the semantics of computationally-oriented languages. 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, Pietro DiGianantonio, Marcelo Fiore, John Power, Alex Simpson

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. Research in type theory itself is also an active concern of this project, with applications not only to logic but also –and more traditionally– to programming languages.

Gordon Plotkin, Andrew Barber, Phillippa Gardner, John Power, Alex Simpson

[78] Brian McConnell and J. V. Tucker. *Editor of Infinite Synchronous Concurrent Algorithms: The Algebraic Specification and Verification of a Hardware Stack*, volume 94 of *NATO ADI Series F: Computer and Systems Sciences*. Springer-Verlag, JUL 1993.

[79] J. McKinna and R. Burstall. Deliverables: A categorical approach to program development in type theory. In *Proceedings of Mathematical Foundations of Computer Science, MFCS'93, Gdansk, Poland*, volume 711, page 36. Springer-Verlag, SEP 1993.

[80] J. McKinna and R. A. Pollack. Pure type systems formalized. In *Proceedings International Conference on Typed Lambda Calculi and Applications, TLCA'93*, volume 664, pages 289–305. Springer-Verlag, MAR 1993.

[81] M. Mendler. *A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification*. PhD thesis, University of Edinburgh, 1993.

[82] A. J. R. G. Milner. Action calculi, or concrete action structrures. In *Proc. MFCS Conference*, volume 711, pages 105–121. Springer-Verlag, SEP 1993.

[83] A. J. R. G. Milner. An action structure for synchronous Pi-calculus. In *FCT Conference, Szeged, Hungary*, volume 710, pages 87–105. Springer-Verlag, AUG 1993.

[84] A. J. R. G. Milner. Action structures for the Pi-calculus. LFCS Report 264, University of Edinburgh, 1993.

[85] A. J. R. G. Milner. Elements of interaction. *Communications of ACM*, 36(1):79–88, 1993.

[86] A. J. R. G. Milner. Modal logics for mobile processes. *Theoretical Computer Science*, 114:23, 1993.

[87] A. J. R. G. Milner. The polyadic Pi-Calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94, pages 203–246. Springer-Verlag, 1993.

[88] F. G. Moller. Cinnamon buns. In ?, page 2. ?, JUL 1993.

[89] F. G. Moller, Y. Hirshfeld, and S. Christensen. Bisimulation equivalence is decidable for basic parallel processes. In *Proceedings of CONCUR'93*. University of Edinburgh, AUG 1993.

[90] F. G. Moller and A. J. R. G. Milner. Unique decomposition of processes. *Journal of Theoretical Computer Science*, 107:7, JAN 1993.

[91] M. Norman and P. Thanisch. Models of machines and computation for mapping in multicomputers. *ACM Computing Surveys*, 25(3):263–302, SEP 1993.

[63] S. M. Kahrs. Pannda standard types and predefined type schemata, MAR 1993.

[64] S. M. Kahrs, D. T. Sannella and A. Tarlecki. The semantics of Extended ML: A gentle introduction. In *Procedings of Int. Work. on Semantics of Specification Languages*, pages 186–215. Springer Verlag, OCT 1993.

[65] K. Kalorkoti. Inverting polynomials and formal power series. *Society for Industrial and Applied Mathematics*, 22(3):8, JUN 1993.

[66] G. Kelly and J. Power. Adjunctions whose counits are coequalisers, and presentations of finitary enriched monads. *Journal of Pure and Applied Algebra*, 89:163–179, 1993.

[67] T. Kelly and R. N. Ibbett. Parallelism versus performance - matching parallel hardware to software. In *Proc. International Conference on Parallel Computing, ParCo'93*, page 9. Elsevier, SEP 1993.

[68] Y. Lee, D. Lee, and A. S. Wight. Performance modelling and simulation of the 1993 Daejon international EXPO network. In *Proceedings of UKSS'93*, pages 23–29. United Kingdom Simulation Society, SEP 1993.

[69] M. Lekuse and D. T. Sannella (translators). *Deduction: Automated Logic by W. Bibel)*. Academic Press, 1993.

[70] C. Leopold. Locality and models of parallel computation, SEP 1993.

[71] C. Leopold and T. H. Heywood. Models of parallelism. In *Proceedings of 2nd Workshop on Abstract Machine Models for Highly Parallel Computing*, page 13. Oxford University Press, 1993.

[72] Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. International Series of Monographs on Computer Science. Oxford University Press, MAR 1993.

[73] Z. Luo. Program specification and data refinement in type theory. Technical report, University of Edinburgh, 1993.

[74] Z. Luo. Program specification and data refinement in type theory. *Mathematical Structures in Computer Science*, 3, 1993.

[75] S. Maharaj. Encoding Z schemas in type theory, 1993.

[76] S. Maharaj and E. Gunter. Studying the ML module system in HOL. Technical Report Technical Report, AT&T Bell Labs, 1993.

[77] S. Manoharan. *Task Assignment in Parallel Processor Systems*. PhD thesis, University of Edinburgh, 1993.

Object-Oriented Database Systems

In the past, database systems were passive repositories of data, limited in use to areas such data processing. Object-oriented database systems have made it possible to apply database technology to new areas of computing. For example, we are using object-oriented database systems in two research project s in the Department, namely medical imaging and architecture simulation. These new application areas have created a number of research issues. In particular , we are investigating the possibilities for exploitation of the advanced transaction processing capabilities of object-oriented databases to support distributed applications in a client/server computing environment.

Peter Thanisch

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 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

It is difficult for programmers to utilize parallel machines effectively unless they are prepared to learn a considerable amount about the idiosyncrasies of each new machine and spend time tuning up their programs for that particular environment. The resulting programs are then not necessarily portable to a new environment. Apart from the waste of time, the overheads involved in using parallel machines have worked against their widespread acceptance by the general community of applications programmers.

The aim behind the work described here is to underpin the design of operating systems that would remove some of the burden from programmers by supporting run time load-balancing. The hope would be that generally good performance could be achieved, even if the performance was not optimal for any particular program. However, to provide a firm basis for design, it is first necessary to understand more about the run-time behaviour of parallel programs; to see whether there are similarities, from a statistical point of view in the behaviour of various types of programs, and to study the evolution of programs in time. 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. These studies show that it is feasible to interpret performance in terms of a very simple performance model, which can be extended to describe time-varying behaviour.

Work is currently proceeding in a number of directions: extending the empirical models to include parameters describing the machine environment, studying the relation between probabilistic changes on individual processors and what is observed on a macroscopic scale, and looking at the requirements for the design of operating systems that are both provably correct and support good user program performance. 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

[50] Y. Hirshfeld, F. G. Moller, and S. Christensen. Decomposability, decidability and axiomatizability for bisimulation equivalence on basic parallel processes. In *Proceedings of the Eighth LICS Symposium*, page 11. IEEE Computer Society Press, 1993.

[51] A. Hondroudakis and R. N. Procter. Abstract machine models and parallel software performance analysis. In *Proceedings of the Second Workshop on Abstract Machine Models for Highly Parallel Computers*, APR 1993.

[52] H. Huttel. Undecidable equivalences for basic parallel processes. LFCS Report 276, University of Edinburgh, 1993.

[53] K. Imre. Experiences with monitoring and visualising the performance of parallel programs. In *Performance Measurement and Visualisation of Parallel Systems*, volume 7. Elsevier Science Publishers B.V., OCT 1993.

[54] K. Imre. *A Performance Monitoring and Analysis Environment for Distributed Memory MIMD Programs.* PhD thesis, University of Edinburgh, 1993.

[55] P. Jancar. Decidability questions for bisimilarity of Petri nets and some related problems. LFCS Report 261, University of Edinburgh, 1993.

[56] M. R. Jerrum. An analysis of a Monte Carlo algorithm for estimating the permanent. In *Proceedings of the 3rd Conference on Integer Programming and Combinatorial Optimization*, pages 171–182. CORE, Louvain-la-Neuve, Belgium, 1993.

[57] M. R. Jerrum. Uniform sampling modulo a group of symmetries using markov chain simulation. LFCS Report 272, University of Edinburgh, 1993.

[58] M. R. Jerrum. Uniform sampling modulo a group of symmetries using markov chain simulation. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 10, pages 37–47. AMS, 1993.

[59] M. R. Jerrum and G. B. Sorkin. Simulated annealing for graph bisection. LFCS Report 260, University of Edinburgh, 1993.

[60] M. R. Jerrum and G. B. Sorkin. Simulated annealing for graph bisection. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, pages 94–103. Computer Society Press, 1993.

[61] S. M. Kahrs. Compilation of combinatory reduction systems. In *Proceedings of Higher-Order Algebra, Logic and Term-Rewriting*, pages 169–188. Springer, SEP 1993.

[62] S. M. Kahrs. Mistakes and ambiguities in the definition of Standard ML. LFCS Report 257, University of Edinburgh, 1993.

[36] K. Goossens. The formalisation of a hardware description language in a proof system: Motivation and applications. In *Proceedings of the XIII Conference of the Brazilian Computer Society*, SEP 1993.

[37] K. Goossens. Structure and behaviour in hardware verification. In *Higher Order Logic Theorem Proving and Its Applications*, AUG 1993.

[38] K. Goossens. Structure and behaviour in hardware verification. LFCS Report 273, University of Edinburgh, 1993.

[39] R. Gordon and J. Power. Enrichment through variation. LFCS Report 254, University of Edinburgh, 1993.

[40] T. Harris and M. I. Cole. The parameterised PRAM. In *Proceedings of the Workshop on Parallel and Distributed Processing, Sofia (Boyonov, Ed.)*, page 10. Elsevier, MAY 1993.

[41] T. Harris and N. P. Topham. Performance of weak consistency schemes on the DEC Alpha. In *Proc. International Conference on Parallel Computing ParCo'93*, page 10. Elsevier, SEP 1993.

[42] C. Hermida. *Fibrations, Logical Predicates and Indeterminates*. PhD thesis, University of Edinburgh, 1993.

[43] R. Hexsel and N. P. Topham. The performance of parallel loops on SCI-based memory hierarchies. In *Proceedings PARCO'93*, page 4. ?, SEP 1993.

[44] T. H. Heywood and C. Leopold. Models of parallelism. CS Report 28, University of Edinburgh, 1993.

[45] T. H. Heywood, K. Mehrotra, and S. Ranka. The performance of Barnes-Hut N-body simulation in the abstract. CS Report 27, University of Edinburgh, JUL 1993.

[46] J. Hillston. An enhanced process algebra for performance modelling. In *Proceedings of the UK Performance Engineering Workshop*, pages 3–14. Loughborough University of Technology, JUL 1993.

[47] J. Hillston. PEPA: Performance enhanced process algebra. CS Report 24, University of Edinburgh, 1993.

[48] J. Hillston. A tool to enhance model exploitation. In *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 10, page 12. Edinburgh University Press, AUG 1993.

[49] J. Hillston and F. G. Moller. Proceedings of the workshop on process algebra and performance modelling. CS Report 26, University of Edinburgh, MAY 1993.

multiprocessor machine in the department.

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

Quantitative System Evaluation

Quantitative system evaluation is concerned with the capture and analysis of the dynamic behaviour of systems such as computers, communication networks and flexible manufacturing systems. This involves the investigation of the flow of data, control or goods within and between components of the system. The aim is to understand the behaviour of the system and identify the aspects of the system which are sensitive from a quantitative point of view.

The size and complexity of many modern systems result in large, complex models. This has led to a renewed interest in simulation techniques and has stimulated new work in the areas of compositional approaches to model construction; model simplification and state space reduction techniques; and efficient solution algorithms.

Simulation techniques

Work in improving the effectiveness of simulaiton modelling concentrates on easier description of simulation models, especially use of graphical and hierarchical formalisms, on efficient exploitation of simulation (and potentially other) models, through the use of object oriented database technology to control experimentation, and increasing integration with qualitative methods, such as protocol specification

languages and process algebras (see below). These approaches are exploited to build novel tools for modelling.

Stochastic process algebras

Historically, quantitative analysis and investigation of the temporal properties of systems has been distinct from qualitative analysis which aims to establish the functional behaviour of systems. However, recent work with stochastic process algebras and the use of pure process algebra to study simulation models allow these important approaches to integrated.

Stochastic process algebras such as PEPA also offer exciting new approaches to the manipulation and simplification of Markov processes. The exploitation of the structure inherent in such models for aggregation of the underlying state space has already been demonstrated and the exploitation of this structure for efficient solution techniques is currently being researched.

Applications of performance modelling

Application of performance analysis techniques is a central aspect of computer systems research. Recent examples include study of parallel program execution using discrete event simulation, design of efficient process migration strategies using specially built monitoring hardware and modelling of heterogeneous computer networks linked by FDDI rings.

Rob Pooley, Alex Wight, Jane Hillston

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

[24] M. Dyer, A. Frieze, and M. R. Jerrum. Approximately counting hamilton cycles in dense graphs. LFCS Report 259, University of Edinburgh, 1993.

[25] J. Esparza. On the decidability of model checking for several mu-calculi and Petri nets. LFCS Report 274, University of Edinburgh, 1993.

[26] M. Fiore. A coinduction principle for recursive data types based on bisimulation. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, volume 8, pages 110–119. IEEE Computer Society Press, JUN 1993.

[27] M. Fiore. Cpo-categories of partial maps. In *?*, pages 45–49. CWI, SEP 1993.

[28] P. Gardner. A new type theory for representing logics. In *Logic Programming and Automated Reasoning*, volume 698, pages 146–157. Springer-Verlag, MAY 1993.

[29] L. Goldberg, M. R. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures*, pages 300–309. ACM, 1993.

[30] L. Goldberg, M. R. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. Nec research institute technical report 93-016-3-0054-3, NEC, JAN 1993.

[31] P. Goldberg. *PAC-Learning Geometrical Figures*. PhD thesis, University of Edinburgh, 1993.

[32] P. Goldberg and M. R. Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterised by real numbers. In *Proceedings of the 6th ACM Symposium on Computational Learning Theory*, page 10. ACM, 1993.

[33] P. Goldberg and M. R. Jerrum. Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. Nec research institute technical report 93-028-3-9004-1, NEC, FEB 1993.

[34] K. Goossens. *Embedding Hardware Description Languages in Proof Systems*. PhD thesis, University of Edinburgh, 1993.

[35] K. Goossens. The formalisation of a hardware description language in a proof system: Motivation and applications. LFCS Report 269, University of Edinburgh, 1993.

[10] G. Brebner. A CCS-based investigation of deadlock in a multi-process electronic mail system. *Formal Aspects of Computing*, 3(5):467–479, OCT 1993.

[11] G. Brebner. Configurable array logic circuits for computing network error detection codes. *Journal of VLSI Signal Processing*, 6(2):101–117, AUG 1993.

[12] G. Bruns. A practical technique for process abstraction. In *Proceedings of CONCUR '93*, pages 37–49. Springer Verlag, AUG 1993.

[13] G. Bruns and S. O. Anderson. Validating safety models with fault trees. In *Proceedings SAFECOMP '93*, pages 21–30. Springer Verlag, OCT 1993.

[14] E. R. S. Candlin and J. G. Phillips. Statistical modelling as a tool for studying the performance of parallel programs. In *Proceedings of the Leeds Workshop on Abstract Machine Models*, page 23. Oxford University Press, APR 1993.

[15] E. R. S. Candlin and J. G. Phillips. A statistical study of the factors that affect the performance of a class of programs on a MIMD computer. In *Proceedings of the International Conference on Decentralized and Distributed Systems (ICDDS'93)*, pages 175–186. IFIP/North-Holland, SEP 1993.

[16] P. Cenciarelli and E. Moggi. A syntactic approach to modularity in denotational semantics. In *CTCS-5, Amsterdam*, SEP 1993.

[17] L. Chen. *Timed Processes: Models, Axioms and Decidability*. PhD thesis, University of Edinburgh, 1993.

[18] S. Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, University of Edinburgh, 1993.

[19] M. I. Cole. List homomorphic parallel algorithms for bracket matching. CS Report 29, University of Edinburgh, 1993.

[20] M. I. Cole. Parallel programming, list homomorphisms and the maximum segment sum problem. CS Report 25, University of Edinburgh, MAY 1993.

[21] M. I. Cole. Parallel programming, list homorphisms and the maximum segment sum problem. In *Proceeding of PARCO '93, Grenoble (Trystam, Ed.)*, page 4. Elsevier, SEP 1993.

[22] A. Compagnoni and B. C. Pierce. Multiple inheritance via intersection tyoes. LFCS Report 275, University of Edinburgh, 1993.

[23] A. J. C. Duggan. Literate programming: A review. Cs dept. report, University of Edinburgh, 1993.

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. The Computer Systems Group (CSG) is a more recently established unit concentrating on the engineering aspects of Computer science both in the hardware and software fields. With the passing of many traditional areas of research into the commercial domain, the formation of this new group has allowed a re-focusing of the general research work of the department. New links with North American and European research groups are being formed and research programs established.

Computer Systems Group

The CSG is a focus for research into all aspects of computer systems. This includes research into a diverse range of computer science topics where performance or

capability of a potential computing system can be seen as the end objective.

The group takes its strength from the long history of computer development at Edinburgh that has seen programming languages, operating systems and workstations developed and used both inside and outside this department. With the commercialisation of these fields due to the greater complexity of achieving worthwhile objectives in these areas, the aim of this research has changed. Of particular significance is the work being carried out both in developing and using simulation environments which provide visual feedback to improve the capability and performance of a range of computer related problems.

There are many systems related problems with parallel machines, which fostered by the strong links with the Edinburgh Parallel Computing Centre, form the basis of a lot of the research interest here.

With the formation in the summer of 1994 of the Scottish Centre of the Mcleod Institute of Simulation Sciences, part of a network of centres of excellence in simulation in Europe and North America, am inportant new inter-disciplinary aspect has been added to the role of the CSG."

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 dependable 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 and is also coordinating node for a European nework of excellence which aims to cohere different hardware

Publications

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

Publications

References

[1] T. Altenkirch. *Constructions, Inductive Types and Strong Normalization.* PhD thesis, University of Edinburgh, 1993.

[2] T. Altenkirch. A formalization of the strong normalization proof for System F in LEGO. In *Typed Lambda Calculus and Applications*, volume 644, page 16. Springer-Verlag, MAR 1993.

[3] P. Anderson. Software installation on large systems. *;login:*, pages 18–20, MAR 1993.

[4] S. O. Anderson and G. Bruns. The formalization and analysis of a communication protocol. *Formal Aspects of Computer Science*, 1993.

[5] I. Androutsopoulos, G. Ritchie, and P. Thanisch. MASQUE/SQL - An efficient and portable natural language query interface for relational databases. In *The Proceedings of the Sixth International Conference on Industrial and Enginering Applications of Artificial Intelligence and Expert Systems*, pages 327–330. Gordon and Breach, JUN 1993.

[6] I. Androutsopoulos, G. Ritchie, and P. Thanisch. MASQUE/SQL - An efficient and portable natural language query interface for relational databases. Department of artificial intelligence research report, University of Edinburgh, 1993.

[7] D. K. Arvind. Locally optimistic methods of concurrent simulation. In *Proceedings of 1993 European Design Automation Conference.* IEEE Computer Society, SEP 1993.

[8] D. K. Arvind. Parallelisation of time-delay neural networks–analytical models and experimental results. In *Proceedings of 1993 Conference on Parallel Computing.* Elsevier, 1993.

[9] J. C. Bradfield. A proof assistant for symbolic model-checking. In *Computer Aided Verification, 1992, Proceedings*, volume 663, page 14. Springer-Verlag, FEB 1993.

Effective software support for chemical research

Amanda Welsh

Abstract: Locating metal bound hydride ligands in transition metal cluster compounds can be difficult with conventional analytical chemistry. Thus many empirical and theoretical methods have been developed for locating such ligands. One such method has been implemented in the Complete Coordinate Convergence Program (CCCP). However, on close examination, this tool is found to be user-friendly and its success is shown to be dependent on an initial location estimate from the user.

Thus the locator was developed as a tool designed to overcome these problems. Methods are presented which exploit known chemical and spatial restraints in finding an initial estimate of the hydrride ligand position(s) for more reliable subsequent optimisation by the CCCP code. The performance of Locator was tested on 178 models, with a favourable success rate of 72%. This figure rises to 84% when taking account of bonding interactions of the hydride ligand(s) provided from the user.

These improvements to CCCP were motivated by a user requirements analysis undertaken to identify computational chemistry applications which would benefit from the exploitation of software engineering and HCI principles becoming prevalent in mainstream computer science. A molecular graphics tool was also selected because it was considered that this would be useful to many researchers, and because the available molecular graphics tools were observed to be lacking in terms of human-computer interaction principles. Further, such a tool could be used as a base for other applications.

This thesis by no means suggests that the scope of physical scientific applications which would benefit from the uptake of such ideas is limited to this relatively narrow field of chemistry. Rather these tools are considered in a *proof of concept* case study within the time restraints of this work.

The adoption of the Visualisor interface and the addition of the initial estimate routine are considered to have significantly improved hydride ligand location in transition metal clusters both in terms of success and accuracy of the results and the ease with which these results may be obtained. Further, it is considered that this project has demonstrated the benefits of the application of current computer science software development principles to scientific software.

software and human reliability issues. 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, 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, 20 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, the Centre for Cognitive Science and the Human Communications Research Centre).

The Laboratory is managed on a day to day basis by its Directorate: Prof. Rod Burstall, Director; Dr. George Cleland, Assistant Director; and Profs. Mike Fourman, Robin Milner, Gordon Plotkin and Dr Don Sannella, 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 project 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.

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 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

CST-109-94

A Statistical Investigation of the Factors Influencing the Performance of Parallel Programs, with Application to a Study of Process Migration Strategies

Joseph Phillips

Abstract: It would be highly desirable for operating systems to take a greater responsibility for process placement and load balancing decisions in parallel machines. This would relieve the programmer of much of the burden associated with fine-tuning an application in order to achieve acceptable performance levels. Before such operating systems can be developed, it is necessary to gain a better understanding of the factors that influence program performance. In particular, it would be useful to be able to identify *classes* of programs which, in a statistical sense, behaved in a similar manner. Then, given an arbitrary program whose class was known, rules and heuristics developed for the program class (in conjunction with program specific information) could be used to make informed placement and load balancing decisions. As a step in this direction, this thesis investigates the application of standard statistical techniques to the performance analysis of particular classes of parallel programs.

Simple CSP-type parallel programs exhibiting loosely synchronous data parallelism are used to illustrate how a common class of programs can be characterised in terms of a relatively small number of parameters representing time-averaged properties. In order to systematically explore parameter space, synthetic programs are used. The execution of these programs is simulated on an accurate performance model of a transputer-based machine. Standard experimental design techniques, such as the analysis of variance, are then applied to develop sstatistical models relating to the program class. It is shown that useful quantitative predictions can be made for arbitrary class members.

The accuracy of the performance model described above can be improved by taking account of run-time information. The analysis of covariance is a technique which enables this by allowing one to incorporate a number of *covariates* into a model. The covariates investigated in this thesis relate to the dynamic properties of programs. More specifically, they are a product of the complex interactions which occur at run-time between a program and the underlying machine.

A representative process migration strategy of the type that might be incorporated into an operating system is presented. The covariate-based performance model developed earlier is then used to identify a number of performance measurements which, when optimised, tend to result in improved processor utilisation. A modified migration strategy which makes migration decisions with these optimisations in mind is presented. It is shown that the new strategy can offer significant performance advantages over the original strategy.

Finally, in order to demonstrate the generality of the new process migration strategy, it is tested on a class of dynamically varying programs. Statistical techniques are used to identify the circumstances under which the strategy can offer the greatest performance benefits. The results obtained from the simulation system are validated by showing that they apply to real programs running on a real transputer-based machine.

CST-108-94

Adding Safe and Effective Load Balancing to Multicomputers

Paul Martin

Abstract: In the quest for ever greater cost-effectiveness researchers have begun to experiment with scalable, parallel architectures known as 'multicomputers'. The underlying assumption is that adding more processors to a computer is a cheap way to increase the problem size which it can tackle and/or decrease the execution time. However, results to date are less good than those hoped for, indicating that there are still a number of difficulties to be resolved. One problem in particular is felt by experienced multicomputer programmers looking for significant execution time speed-ups: much effort must be expended to tune a program for the underlying architecture if the work is to be evenly distributed between the processors. Fortunately, a solution to this problem can be found in dynamic load balancing, a mechanism for redistributing work between processors automatically and transparently, allowing the programmer to develop fast, portable programs without having to worry about performance tuning.

This thesis examines the many issues associated with adding load balancing to multicomputers and makes the following contributions. Firstly, a critical review of the literature on load balancing showing the techniques proposed and suggesting which are the most promising for future systems. Secondly, a detailed description of how a typical multicomputer operating system needs to be extended in order to be able to checkpoint a task on one processor and restart it on another. Thirdly, a study of the impact on performance of these extensions to the operating system functionality. Fourthly, an investigation into the use of formal methods for designing and verifying the protocols for exchange of tasks between processors. Lastly, a report on the best methods for detecting processor load imbalances and for deciding which user tasks to move. Thus, the thesis addresses all aspects of adding safe and effective load balancing to multicomputers.

PhD Thesis – Price £8.50

research themes above.

The Laboratory publishes a widely acclaimed report series comprising both research and expository material. Copies of these and other publications are available both by FTP and on the world wide web, or may be obtained directly from George Cleland, the Assistant Director, from whom further general information on the Laboratory may also be obtained.

Contact Information for LFCS:

LFCS, Dept of Computer Science, The University of Edinburgh, Edinburgh EH9 3JZ, UK. tel:(+44) 0131-650-5199. Fax: (+44) 0131-667-7209

E-mail:`lfcs@ed.ac.uk`

LFCS Software and reports are available by anonymous FTP:
`ftp.lfcs.ed.ac.uk`

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

Reports and software, as well as much detailed information about LFCS is available on the World Wide Web:
`http://www.dcs.ed.ac.uk/lfcs`

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.

ΔΔΔΔ

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.

CST-107-94

A Compositional Approach to Performance Modelling

Jane Hillston

Abstract: Performance modelling is concerned with the capture and analysis of the dynamic behaviour of computer and communication systems. The size and complexity of many modern systems result in large, complex models. A compositional approach decomposes the system into subsystems that are smaller and more easily modelled. In this thesis a novel compositional approach to performance modelling is presented. The approach is based on a suitably enhanced process algebra, PEPA (Performance Evaluation Process Algebra). The compositional nature of the language provides benefits for model solution as well as model construction. An operational semantics is provided for PEPA and its use to generate an underlying Markov process for any PEPA model is explained and demonstrated. Model simplification and state space aggregation have been proposed as means to tackle the problems of large performance models. These techniques are presented in terms of notions of equivalence between modelling entities.

A framework is developed for anlysing such notions of equivalence and it is explained how the bisimulation relations developed for process algebrs fit within the framework. Four different equivalence relations for PEPA, two structural and two based on bisimulation, are developed and considered within this framework. For each equivalence the implications for the underlying Markov process are studied and its potential use as the basis of a model simplification technique is assessed. Three of these equivalences are shown to be congruences and all are complementary to the compositional nature of the models considered. As well as their intrinsic interest from a process algebra perspective, each of these notions of equivalence is also demonstrated to be useful in a performance modelling context. The strong structural equivalence, *isomorphism*, generates equational laws which form the basis of model transformation techniques. This is weakened to define *weak isomorphisms*. This equivalence, together with judicious use of the PEPA abstraction mechanisms, forms the basis of a model simplification technique, provided certain insensitivity conditions are satisfied. *Strong bisimilarity* is shown to exhibit no clear relationship to the underlying Markov process although it may be used to replace one component of a model by another which will have the same apparent behaviour. Finally, *strong equivalence*, provides an alternative method of formulating the Markov process capturing the stochastic behaviour of the model. This equivalence is the basis of an aggregation technique based on lumpability.

Throughout the thesis the concepts introduced are illustrated by examples modelling *multi-server multi-queue* (MSMQ) systems. These systems, an extension of classical polling systems, have been shown to be useful representations of many local area network architectures, with ring topologies and scheduled access, in which more than one code may transmit simultaneously.

PhD Thesis – Price £8.50

Supported Projects

Title	Holder	Value	Funding Body	Start	End
Declaritive Languages and Applied Semantics	Milner, Fourman, Mitchell	£391,230	EPSRC	1-Apr-90	31-Mar-94
Logical and Semanical Frameworks	Plotkin, Moggi	£329,035	EPSRC Rolling	1-Oct-90	30-Mar-95
Constructive Logic as a Basis for program Development	Burstall	£336,192	EPSRC Rolling	1-Jan-91	30-Jun-95
Verification of Infinite State Systems	Stirling	£85,513	EPSRC	1-Oct-91	31-Mar-95
Infusion of concurrency and comm. into programs	Lengauer, Arvind	£106,693	EPSRC	1-Jun-91	30-Nov-94
Quantitive analysis of stochastic systems	Sinclair, Jerrum	£93,072	EPSRC	1-Jul-91	31-Dec-94
Formal Reasoning Awareness Club	Cleland	£25,500	DTI Contract	1-Jul-91	30-Jun-95
Types, Proofs and Programs	Burstall, Plotkin	£234,646	CEU, BRA	1-Sep-92	31-Aug-95
Concurrency and Functions: Evaluations and Reductions	Milner	£118,110	CEU, BRA	1-Sep-92	31-Aug-95
Calculi and algebras of Concurrency Extensions & Applications	Stirling, Milner	£56,393	CEU, BRA	1-Sep-92	31-Aug-95
Typed Lambda Calculus Network	Plotkin	£58,333	CEU HCM	1-Sep-93	31-Aug-96
Programming Language semantics and program logics	Plotkin, Moggi	£59,500	CEU Science	1-Sep-93	31-Aug-96
Algebraic and logical foundations of formal SW devment	Sannella	£116,625	EPSRC	16-Feb-93	15-Feb-96
A compreh. Alg. Approach to System Spec and Devment	Sannella, Burstall	£15,748	CEU ESPRIT	1-Sep-92	31-Aug-93
Randomised Algorithms	Jerrum	£15,358	CEU ESPRIT	1-Sep-92	31-Aug-95
Randomness and Computation	Jerrum	£16,680	EPSRC	1-Jun-93	30-Sep-93
Foundations of Formal System Development	Sannella	£16,357	CEU ESPRIT	1-Sep-93	31-Aug-96
Sannella SERC Advanced Fellowship - Computing Support	Sannella	£7,434	EPSRC	1-Jan-93	31-Dec-93
Formal dvpt of Modular Programs in Extended ML	Sannella	£142,409	EPSRC	16-Feb-93	16-Feb-96
VMS File System Project	Arvind	£17,500	DEC	6-Feb-93	7-Jun-94
Communication in Safety Cases - A Semantic Approach	Anderson, Clelan	£586,928	EPSRC	1-Jul-93	30-Jun-96
Development of High Integrity Systems	Anderson, MacKenzie[2]	£139,118	ESRC/SERC	1-Jan-93	30-Sep-96
Fellowship: Le Sergent	Fourman	£43,046	CEU Esprit	1-Oct-93	30-Sep-94
Fellowship: Hofmann	Sannella	£57,552	CEU Esprit	1-Aug-93	31-Jul-95
Fellowship: Tarlecki	Sannella	£6,240	CEU Esprit	1-Jan-93	30-Dec-94
Distributed Shared Environments for Concurrent ML	Fourman, Mitchell, Ibbett	£194,495	EPSRC	1-Feb-94	31-Jan-97
Teaching Company Scheme with Adelard	Cleland, Anderson	£166,619	DTI/EPSRC/TCS	21-Jun-93	20-Jun-96
EuroFOCS - European Network in Logical Foundations of CS	Plotkin	£100,000	CEU HCM	1-Dec-93	30-Nov-96
EUROFOCS - Fellowships, phase I	Plotkin	£417,000	CEU HCM	1-Jan-94	30-Dec-96
EUROFOCS - Fellowships, phase II	Plotkin	£180,000	CEU HCM		
EuroFOCS - East Europe Extension	Plotkin	£60,000	CEU HCM		
Frameworks for Programming Language Semantics and Logic	Plotkin	£294,691		1-Apr-95	31-Mar-98
Applications of a Type Theory Based Proof Assistant	Burstall	£377,340		1-Apr-95	31-Mar-98
Applied Formal Methods Feasibility Study	Cleland, Anderson	£38,000	DTI	1-Oct-93	30-Dec-95
Frameworks for Programming Language Semantics and Logic	Plotkin	£294,691		1-Jan-95	
Applications of a Type Theory Based Proof Assistant	Burstall	£377,340		1-Oct-94	
	Gardner		BP/RSE	1-Jan-94	30-Dec-95
	Simpson		SERC		
An Holistic Approach to System Reliability	Cleland, Anderson	£870,000	CEU HCM	1-Jan-95	31-Dec-98

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 and non-profit organisations 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.

The proof of completeness relies on the unique decomposition property admitted by this equivalence on BPP.

PhD thesis – Price £8.50

CST-105-93 (also published as LFCS-93-278)

Decidability and Decomposition in Process Algebras

Søren Christensen

Abstract: This thesis is concerned with the question of obtaining decidable theories for behavioural equivalences on various models of (parallel) computation encompassing systems with infinitely many states. The models on which we concentrate are based on the process calculi BPA and BPP but we also consider labelled Petri nets. The equivalences which we are interested in are language equivalence, bisimulation equivalence and distributed bisimulation equivalence.

BPA (Basic Process Algebra) is provided by a standard calculus which admits of a general sequencing operator, along with atomic actions, choice and recursion. In contrast, BPP (Basic Parallel Processes) is provided by a standard calculus which admits of a simple parallel operator (full merge), along with atomic actions, choice and recursion. From the point of view of language equivalence we show that BPA and BPP are incomparable. This result is in part obtained through a pumping lemma for BPP. We furthermore show that the class of languages generated by BPP is included in a class of languages generated by labelled Petri nets as well as contained in the class of context-sensitive languages. Next we investigate into decidability of language equivalence on language classes related to BPP and Petri nets.

We then move on to bisimulation equivalence. We show that this equivalence is decidable on all of BPA, answering a question left open by Baeten, Bergstra and Klop. We also show that bisimulation equivalence is decidable on BPP and BPP_r, (where BPP_r is similar to BPP except for its parallel operator which allows for synchronisation). By these results we have obtained a delicate line between decidable and undecidable theories for bisimilarity; by extending BPP_r, with the operator of restriction we know that bisimilarity is undecidable. By relying on Jančar's recent result on the undecidability of bisimilarity on labelled Petri nets we further narrow the gap between decidable and undecidable theories; when extending BPP with a notion of forced synchronisation we know that bisimilarity is undecidable.

The decidability proofs of bisimulation equivalence on BPP and BPP_r (obtained via the tableau technique) permit us further to present sound and complete equational theories for these process classes. As BPP and BPP_r contain the regular processes their results may be seen as proper extensions of Milner's equational theory for regular processes.

In this thesis we shall also consider the problem of decomposing a process into the parallel composition of simpler processes. A particular aspect of decomposition involves decomposing uniquely into prime processes, i.e. those processes which cannot themselves be expressed as a parallel composition of other non-trivial processes. We shall present several unique decomposition results for bisimilarity and distributed bisimilarity on BPP and BPP_r as well as subclasses thereof.

Finally, for distributed bisimulation equivaalence we show decidability on the process classes BPP and BPP_r. Again our proofs of decidability permit us to present sound and complete equational theories. These results may be seen as extensions of Castellani's equational theories for distributed bisimilarity on the recursion-free fragments of BPP and BPP_r. Also for a fragment of BPP_r where general summation is replaced by guarded summation shall we present an equational theory for distributed bisimilarity.

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.

Shell Expro, Praxis, British Rail, AEA Technologies, Lucas AutomotiveAdelard, H 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.

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

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

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 1995.

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-106-93 (also published as LFCS-93-279)

Constructions, Inductive Types and Strong Normalization

Thorsten Altenkirch

Abstract: This thesis contains an investigation of Coquand's Calculus of Constructions, a basic impredicative Type Theory. We review syntactic properties of the calculus, in particular decidability of equality and type-checking, based on the equality-as-judgement presentation. We present a set-theoretic notion of model, CC-structures, and use this to give a new strong normalization proof based on a modification of the realizability interpretation. An extension of the core calculus by inductive types is investigated and we show, using the example of infinite trees, how the realizability semantics and the strong normalization argument can be extended to non-algebraic inductive types. We emphasize that our interpretation is sound for *large eliminations*, e.g. allows the definition of sets by recursion. Finally we apply the extended calculus to a non-trivial problem: the formalization of the strong normalization argument for Girard's System F. This formal proof has been developed and checked using the LEGO system, which has been implemented by Randy Pollack. We include the LEGO files in the appendix.

PhD thesis – Price £8.00