

Computer Science
Departmental Report 1991-92

Contents

Introduction	3
Members of Staff 1992-93	5
Undergraduate Courses	14
Postgraduate Study	17
Postgraduate Students 1992-93	19
Departmental Computing Facilities	25
Research Interests	27
Laboratory for the Foundations of Computer Science (LFCS)	42
The Edinburgh Parallel Computing Centre	46
Publications 1992	48
Internal Reports & Theses 1991-92	57
Grant Awards 1991-92	63
Project Summary 1992	64
Industrial Collaborators and Affiliates 1991-92	65

Ford Motor Company

GEC Ferranti

Hewlett Packard

Hitachi Europe Ltd

ICL

Intelligent Applications Ltd

Logica

Spider Systems

Wolfson Microelectronics

Motorola

Siemens (Munich) funds two postgraduate students.

Hewlett Packard funds one overseas postgraduate student.

Computer Science 4

Sun Microsystems : Prize for Best Single Honours student.

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

Computer Science 3

FI Group : Prize for Best Single Honours student;

Ford Motor Company : Prize for Best Group Project;

Hewlett Packard : Prize for Computer Science & Electronics Student;

Thorn EMI (CRL) Bursary;

The CS3 System Design Practical has been supported by a number of companies: **Ford, Hewlett Packard, Logica, 3-L Compilers** and **Seel**. Potential advisors include Andersen Consulting 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:

Digital (Scotland)

Introduction

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

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

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

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

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

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

number of different fields including: computational complexity, parallel systems, theory of computation and VLSI design using both formal and informal methods.

The Department is involved with two major research units within the University. The Laboratory for the Foundations of Computer Science is a federation of research projects within the Department unified by their interest in computational theory and its applications. The main areas of interest are proof engineering, languages and programming, concurrency and communication, specification and formal development of systems, and semantics. The Edinburgh Parallel Computing Centre, which was set up jointly by the Department of Computer Science, the Physics Department and EUCS, encompasses the activities of a variety of research groups. It has as one of its main aims the support of applications work by investigating those fundamental aspects of parallel computing which can lead to the development of new and improved systems and user support tools.

The Department is located in the James Clerk Maxwell Building (JCMB), a multi-user building shared by several academic departments including Mathematics, Meteorology, Physics and Statistics together with the main facilities of the Edinburgh University Computing Service (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.

Industrial Collaborators and Affiliates 1991-92

Projects

Abstract Hardware Limited: Collaborative project project on formal system design; SERC/IED funded.

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.

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.

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

New Initiatives

From December 1993, The Department will host a series of seminars, designed to forge closer ties with technology-related companies. The seminars will provide a forum for discussion on current research directions within in the Department and an opportunity for delegates to share common problems and concerns.

From 1993, Salomon Brothers International will offer a bursary to one CS3 student.

Prizes, Bursaries and Studentships

PhD

The following companies currently support CASE students:

Digital (Scotland)

Inmos

Source	Ref	Title	Holder	Start	End	P. months	Value (£)
CEC Science	ST2/0374	Typed Lambda Calculus	Plotkin	1-Mar-88	28-Feb-92	12	30,000
ESPRIT BRA	3264	Algebraic Approach to System Specification	Sannella, Burstall	1-Mar-88	31-Mar-92	60	20,000
ESPRIT BRA	3265	Logical Frameworks - Design Implementation, Expt.	Harpal, Plotkin, Milner	1-Jun-88	31-Mar-92	168	350,760
ESPRIT BRA	3006	Theories of Concurrency: Unification and Extension	Milner, Stirling	1-Sep-88	31-Mar-92	60	91,540
SERC	GR/E 76487	Formal development of programs from specifications	Sannella	1-May-88	30-Apr-93	48	76,742
ESPRIT II	2143	The integrated modelling support environment	Pooley	1-Jun-88	31-Dec-91	72	250,196
SERC	GR/F 32219	Modal and Temporal MC-Calculi	Stirling	1-Oct-88	30-Sep-92	36	68,200
SERC	IED SE/1224	SERC Senior Fellowship	Milner	1-Jan-90	30-Jun-93	36	28,848
SERC/OTI	GR/F 23590	Mathematically Proven Safety Systems	Cleland, Walker, Anderson	1-Jan-90	30-Jun-93	36	72,510
SERC/OTI	GR/F 23590	Formal System Design	Fourman	1-Jan-90	30-Jun-93	36	89,886
SERC	IED/7/1154	Parallel Processing with Intelligent Knowledge bases	Tharisch, Wallace	1-Jan-90	30-Jun-93	182	347,551
SERC	GR/F 91001	Declarative Languages and Applied Semantics	Milner, Fourman, Mitchell	1-Apr-90	31-Mar-93	180	805,000
SERC	GR/F 91001	Declarative Languages and Applied Semantics	Ibbett, Wallace	1-Apr-90	31-Mar-93	180	805,000
Hewlett Packard	B/18534	Novel Architecture Computing Foundation Support	Cleland	1-Jun-90	1-May-93	36	483,000
SERC	GR/F 731199	Equipment Donation	Fourman	1-Jun-90	30-Nov-93	36	81,040
SERC	GR/F 731199	Formally Based Systems Design Tools	Plotkin	1-Jun-90	30-Sep-93	36	110,068
SERC	GR/G 58548	Category Theory in Computer Science	Plotkin, Moggi	1-Oct-90	30-Sep-93	182	358,331
SERC	GR/G 58548	Logical and Semantical Frameworks	Burstall	1-Oct-90	30-Sep-93	182	379,468
SERC	GR/G 55792	Constructive Logic as a Basis for program Development	Stirling	1-Jan-91	31-Dec-93	36	3,450
SERC	GR/G 55471	Quantitative analysis of stochastic systems	Sindelar, Jernum	1-Apr-91	30-Jun-94	36	106,893
SERC	GR/G 55471	Logical Systems and Logical Frameworks	Plotkin	1-Apr-91	31-Mar-94	36	25,000
DTI	GR/G 55457	Formal Inflation of concurrency and comm. into programs	Langauer, Arvid	1-Jun-91	1-Dec-93	2	2,260
SERC	GR/G 55464	Formal Inflation of concurrency and comm. into programs	Cleland	1-Jun-91	30-Aug-91	2	1,950
British Council	2716	Denotational semantics and concurrency	Sannella	1-Jul-91	30-Jun-93	8	34,000
ESPRIT II	GR/H 18058	Verification of Performance of Decoupled Systems (AMUS)	Sannella	1-Sep-91	31-May-92	36	85,513
SERC	GR/H 18058	Architecture & Performance of Decoupled Systems	Stirling	1-Oct-91	30-Sep-93	24	25,000
Royal Exhibition	B/91/RFH/1210	Verification of Infinite State Systems	Plotkin	1-Jan-92	31-Dec-93	24	28,000
SERC	B/91/RFH/1211	Postdoc Fellowship	McKenna	1-Apr-92	31-Mar-94	24	28,000
SERC	B/91/RFH/1210	Postdoc Fellowship	Burstall, Plotkin	1-Sep-92	31-Aug-95	54	234,848
ESPRIT BRA	6453	Types for Proofs and Programs	Milner	1-Sep-92	31-Aug-95	48	118,110
ESPRIT BRA	6454	Concurrency and Functions: Evaluations and Reductions	Stirling, Milner	1-Sep-92	31-Aug-95	48	135,118
ESPRIT BRA	7166	Calculus of Concurrency: Extensions, Tools and Applications	Plotkin, Moggi	1-Sep-92	31-Aug-95	36	30,000
CEC Science	ERB4002P1310	Programming Language semantics and program logics	Sannella, Burstall	1-Sep-92	31-Aug-92	15,748	15,748
ESPRIT BRA	BRA	A complete, Ag. Approach to System Spec and Devment	Jernum, Sindelar	1-Sep-92	31-Aug-92	15,748	15,748
ESPRIT BRA	BRA	Randomized Algorithms	Fourman	1-Oct-92	30-Sep-93	2,725	2,725
British Council	Pro 0278	Formal Inflation of concurrency and comm. into programs	Sannella	1-Oct-92	30-Sep-97	60	110,713
SERC	GR/H 18058	SERC Advanced Fellowship	Plotkin	1-May-93	30-Apr-98	36	27,752
SERC	GR/H 18058	Algebraic and logical foundations of formal SW devment	Sannella	1-Oct-92	30-Sep-93	12	287,717
SERC	GR/H 18058	Algebraic and logical foundations of formal SW devment	Cleland, Anderson	1-Oct-92	30-Sep-95	96	166,818
ESPRIT III	P 8253	Architecture & Performance of Decoupled Systems (SHIPS)	Topham	1-Jan-93	31-Dec-95	96	166,818
EC/NOE	GR/H 18058	Teaching Company Scheme with Adelard	Plotkin	1-Jan-93	31-Dec-95	96	166,818
EC/NOE	GR/H 18058	European Init. in Local Foundations of Computer Science	Plotkin, Burstall, Milner	1-Jan-93	31-Dec-95	96	166,818
EC/NOE	GR/H 18058	Network of Excellence in Typed Lambda Calculus	Sannella	1-Jan-93	31-Dec-95	96	166,818
EC/NOE	GR/H 18058	Network of Excellence in System Specification	Sannella	1-Jan-93	31-Dec-95	96	166,818

Members of Staff 1992-93

Professors

Professor R.M. Burstall (rb@uk.ac.ed.dcs¹): *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): *System specification, verification and synthesis - formal models of digital behaviour and of the design process; temporal and data abstraction as used in digital design; design and implementation of formally-based system-design tools; proof and proof assistants; specification languages and their semantics.*

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

Professor A.J.R.G. Milner (rm): *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): *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 - Postgraduate Admissions and Thesis Proposal Reviews*
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*
Algebraic specification and formal program development, mechanised reasoning, programming methodology and functional programming languages.

¹Subsequent entries provide usernames only

Senior Lecturers

Mr S.O. Anderson (soa): *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; development of theories of particular application areas within type theory (e.g. user interface design).

Mr R.J. Pooley (rjp): *MSc Projects Co-ordinator; CS/Man. Sci. Course Organiser*

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): *CS4 Project Coordinator*

Computer Aided Design tools for VLSI design; silicon compilation from behavioural descriptions; hardware description languages; silicon assembly; novel VLSI architectures; configurable cellular arrays.

Lecturers

Mr D.K. Arvind (dka): *MSc Course Organiser*

Computer vision — perception of 3D scenes; learning in parallel networks; algorithms and environments for parallel computation; parallel and distributed simulation.

Dr J.C. Bradfield (jcb): *MSc Admissions Officer*

Modal logic and model checking for infinite systems; computer-assisted verification techniques.

Dr G.J. Brebner (gordon): *Director of Studies; Schools' Liaison Representative*

Computer networking, protocol specification and verification, distributed computing environments; configurable cellular arrays; computational complexity.

Dr E.R.S. Candlin (rc): *MSc Director of Studies*

Operating systems for MIMD machines, with particular interest in the support of process migration; models and measurement of parallel computations.

Dr M.I. Cole (mic): *CS1 Course Organiser*

The design and implementation of programming languages and frameworks for parallel computers; the use of the Bird-Meertens theory of lists as a foundation for parallel program design and transformation.

Grant Awards 1991-92

The enclosed table lists funded projects active 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 are 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 comes 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 and ESPRIT basic research actions. 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 new initiative just approved is a teaching company scheme with Adelard in the safety-critical area. 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). There are plans to involve the Human Communications Research Centre in work on Safety-Critical Systems.

CST-86-91 : Decidability, Behavioural Equivalences and Infinite Transition Graphs
Hans Hüttel

CST-87-92 : First Order Linear Logic in Symmetric Monoidal Closed Categories
Simon Ambler

CST-88-92 : Ordinal Complexity of Recursive Programs and their Termination Proofs
Matthew V H Fairtlough

CST-89-92 : Efficient Algorithms for Listing Combinatorial Structures
Leslie Ann Goldberg

CST-90-92 : The Formal Synthesis of Control Signals for Systolic Arrays
Jingling Xue

CST-91-92 : A Framework for Binding Operators
Sun Yong

CST-92-92 : Verification in ASL and related specification languages
Jorge Farrés Casaals

CST-93-92 : Representing Logics in Type theory
Philippa Gardner

CST-94-92 : High-level Synthesis of VLSI Circuits
Ping F Yeung

CST-95-92 : Correctness Proofs of Compilers and Debuggers: An Approach Based on Structural Operational Semantics
Fabio Q B da Silva

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

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

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

Dr M.R. Jerrum (mrj): *CS/Maths/Stats Course Organiser*
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, scene rendering, object modelling on various parallel architectures; computer architectures; VLSI design — synthesis and simulation tools.

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

Dr R.N. Procter (rnp): *Director of Studies*
Human Computer Interaction, especially interfaces for real-time control and collaborative tasks. Organisational factors in systems design and design methodologies; social shaping of Information Technology.

Dr A.J. Sinclair (ajs@edu.berkeley.icsi²): Complexity theory, design and analysis of algorithms, randomised computation and probabilistic algorithms, approximation algorithms for combinatorial problems.

²On leave of absence in the USA until October 1993

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

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

Dr P. Thanisch (pt): Parallel computing systems — load balancing; database systems: object-oriented database systems performance analysis; database design.

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): *Director of Studies — non-Graduating Students; IS1 Course Organiser*
Computer systems performance evaluation; flow and congestion control in computer communications networks; workload characterisation.

Honorary Fellows 1992-93

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

Prof. Dr. C. Lengauer (lengauer@de.uni-passau.fmi): Parallelism between theory and practice - formal parallelising compilation, systolic design; semantics of concurrency, mathematics of program construction; automated theorem proving, implementation of formal methods.

(Senior) Research Workers/Postdoctoral Fellows 1992-93

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

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

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

ECS-LFCS-92-237 : Miscellaneous Design Issues in the ML Kit
Nick Rothwell

ECS-LFCS-92-238 : Polymorphic Type Checking by Interpretation of Code
Stefan Kahrs

ECS-LFCS-92-239 : A Case Study in Safety-Critical Design
Glenn Bruns

ECS-LFCS-92-240 : Observational Equivalence and Compiler Correctness
Fabio Q B da Silva

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

CS Reports 1991-92

CSR-18-91 : Graph Reduction and Pipelined Parallel Algorithms
Murray Cole

CSR-19-92 : Design of a Bit-sliced network for a shared-memory multiprocessor system
D J Rogers and R N Ibbett

CSR-20-92 : A Tool to Enhance Model Exploitation
Jane Hillston

CSR-21-92 : Final Report on the DTI-funded Awareness Initiative in Object Oriented Computing
R N Ibbett

CSR-22-92 : Pointers and Multi-Dimensional Arrays
Richard A Eyre-Todd

Theses 1991-92

CST-85-91 : Design and simulation of an MIMD shared memory multiprocessor with interleaved instruction streams
Tom Stiernerling

ECS-LFCS-92-223 : Relational Parametricity and Local Variables
P W O'Hearn and R D Tennent

ECS-LFCS-92-224 : Fixpoints of Büchi automata
Mads Dam

ECS-LFCS-92-225 : Object-Oriented Programming Without Recursive Types
Benjamin C Pierce and David N Turner

ECS-LFCS-92-226 : An Abstract View of Objects and Subtyping
Martin Hofmann and Benjamin C Pierce

ECS-LFCS-92-228 : Formal Development of Functional Programs in Type Theory – A Case Study
Martin Hofmann

ECS-LFCS-92-229 : A Unification Algorithm for the $\lambda\Pi$ -Calculus
David Pym

ECS-LFCS-92-230 : Brewing Strong Normalization Proofs with LEGO
Thorsten Altenkirch

ECS-LFCS-92-231 : Operational Semantics Based Formal Symbolic Simulation
Kees G W Goossens

ECS-LFCS-92-232 : On the π -Calculus and Linear Logic
G Bellin and P J Scott

ECS-LFCS-92-233 : Correctness Proofs of Compilers and Debuggers: an Overview of an Approach Based on Structural Operational Semantics
Fabio Q B da Silva

ECS-LFCS-92-234 : A Sub-logarithmic Communication Algorithm for the Completely Connected Optical Communication Parallel Computer
Leslie Ann Goldberg and Mark Jerrum

ECS-LFCS-92-235 : Functional Compilation from the Standard ML Core Language to Lambda Calculus
Nick Rothwell

ECS-LFCS-92-236 : Parsing in the SML Kit
Nick Rothwell

Dr C.C.M. Jones (ccmj): Mathematical applications of computer assisted formal reasoning, constructive mathematics, semantics.

Mr G. Jones (gxj): Simulation of decoupled architectures.

Dr B. Jay (cbj): *BP/RSE Research Fellow*
Categorical semantics of programming languages, rewriting systems, domain theory, ordered categories.

Dr T. LeSergent (tl): Implementation of concurrent functional programming.

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

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

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

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

bf Dr R. Martlet (rmar): Semantics of programming languages.

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

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

Dr B.C. Pierce (bcp): *On leave of absence from Carnegie Mellon University, USA*
Type-theoretic foundations for object-oriented programming languages; programming languages based upon Milner's π -calculus.

Mr R.A. Pollack (rap): Design of proof systems, type theory, proof theory.

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

Dr D. Pym (dpym): Proof theory, logical frameworks, categorical logic/models, search spaces, logic programming, linear logic, recursion theory.

Dr R. Sayle (ros): Delay-insensitive and self-timed circuit design methodologies; high-level synthesis of asynchronous VLSI circuits.

Mr D. Sangiorgi (sad): Semantics of concurrent processes, particularly the process algebra.

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

Dr R.D. Tennant (rdt): *On leave of absence from Queen's University, Ontario, Canada*
Semantics and design of programming languages and logics.

Computing Officers

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

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

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

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

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

Mr C. Cooke (cc): Staff systems support; GNU emacs.

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

Mr A.J.C. Duggan (ajcd): p/t Computing Officer; p/t PG
TeX; graphics packages and course support; printers, plotters and spooling; DTP.

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

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

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

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

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

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

LFCS-92-208 : An introduction to fibrations, topos theory, the effective topos and modest sets
Wesley Phoa

LFCS-92-209 : Inductive Data Types: Well-ordering Types Revisited
H. Goguen & Zhaohui Luo

LFCS-92-210 : Listing Graphs that Satisfy First Order Sentences
L A Goldberg

LFCS-92-211 : LEGO Proof Development System: User's Manual
Zhaohui Luo and R Pollack

LFCS-92-212 : On Resolution in Fragments of Classical Linear Logic (Extended Abstract)
J Harland and D J Pym

LFCS-92-213 : A proposed categorical semantics for Pure ML
Wesley Phoa and Michael Fourman

ECS-LFCS-92-214 : From term models to domains
Wesley Phoa

ECS-LFCS-92-215 : Building domains from graph models
Wesley Phoa

ECS-LFCS-92-216 : Process-Algebraic Interpretations of Positive Linear and Relevant Logics
Mads Dam

ECS-LFCS-92-217 : CTL* and ECTL* as fragments of the modal μ -calculus
Mads Dam

ECS-LFCS-92-218 : Bisimulation Equivalence is Decidable for all Context-Free Processes
Soren Christensen, Hans Hüttel, Colin Stirling

ECS-LFCS-92-219 : Unlimp - Uniqueness as a Leitmotiv for Implementation
Stefan Kahrs

ECS-LFCS-92-221 : Modal and Temporal Logics for Processes
Colin Stirling

ECS-LFCS-92-222 : Toward formal development of programs from algebraic specifications: parameterisation revisited
Donald Sannella, Stefan Sokolowski and Andrzej Tarlecki

LFCS-91-189 : The Lazy Lambda Calculus in a Concurrency Scenario
D Sangiorgi

LFCS-91-190 : An n-Categorical Pasting Theorem
J Power

LFCS-92-192 : Semantics of Local Variables
R D Tennent and O'Hearn

LFCS-92-193 : Verification of Parallel Systems via Decomposition
J F Groote & F Moller

LFCS-92-195 : Formal Program Development in Modular Prolog: A Case Study
M Read & E Kazmierczak

LFCS-92-197 : Loop Parallelization and Unimodularity
C. Lengauer and M. Barnett

LFCS-92-199 : A Proof Assistant for Symbolic Model-Checking
J C Bradfield

LFCS-92-200 : Interresection Types & Bounded Polymorphism
Benjamin Pierce

LFCS-92-201 : Approximating the Permanent: a Simple Approach
L E Rasmussen

LFCS-92-202 : A Unifying Theory of Dependent Types: the Schematic Approach
Zhaohui Luo

LFCS-92-203 (also published as CST-90-92) : The Formal Synthesis of Control Signals for Systolic Arrays
Jingling Xue

LFCS-92-204 : Toward Formal Development of Programs from Algebraic Specifications: Model-Theoretic Foundations
D. Sannella & A Tarlecki

LFCS-92-205 : Tail Recursion through Universal Invariants
C B Jay

LFCS-92-206 : A Set-theoretic Setting for Structuring Theories in Proof Development
Z. Luo & R. Burstall

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

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

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

Administrative and Secretarial Staff

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

Technical Staff

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

Visiting Scholars 1991-92

- Dr M.I. Alahmadi:** *King Fahad University, Dharan, Saudi Arabia*
Implementation of FPIF (a parallel intermediate form for FP programs) on EPCC's Meiko.
- Dr J Agustí:** *Centre d'Estudis Avançats de Blanes (CEAB), Catalonia, Spain*
Logical foundations of knowledge-based systems.
- Dr A. Avron:** *Tel-Aviv University*
Logical frameworks; non-classical logics (especially linear and relevance logics); database theory.
- Mr A. Białasik:** *Polish Academy of Science, Warsaw*
Algebraic specification of concurrent programs.
- Mrs Chen Shuyun:** *The Institute of Aeronautic Computation Technology, PR China*
Computer architecture; computer simulation.
- Mr G. Ciobanu:** *University of Iasi, Hungary*
Semantics; concurrency.
- Dr Y. Hirshfeld:** *University of Tel-Aviv*
Mathematical Logic and Non Standard Analysis; concurrency; Computer Science Logic.
- Mrs E Hristokova:** *University of Plovdiv, Bulgaria*
On EC TEMPUS programme - visited to gain familiarity with UK teaching methods.
- Dr Y. Kinoshita:** *Electrotechnical Laboratory, Ibaraki, Japan*
Type theory and constructive modular programming.
- Dr A.R. ter Laak:** *University of Amsterdam*
Parallel Simulated Annealing
- Prof. Philip Mulry:** *Colgate University, Hamilton, New York*
Categorical approach to recursion.
- Dr J. Nummenmaa:** *University of Tampere, Finland*
Database systems design.
- Dr W. Phoa :** *University of New South Wales, Kensington, Australia*
Category theory and topos theory; semantics of programming languages.
- Prof D. Schmidt:** *Kansas State University*
Denotational semantics.

Internal Reports & Theses 1991-92

LFCS Reports⁶

- LFCS-91-178 : Improved bounds for mixing rates of Markov chains and multicommodity flow**
Alistair Sinclair
- LFCS-91-179 : A Mildly Exponential Approximation Algorithm for the Permanent**
M. Jerrum
- LFCS-91-180 : The Polyadic π -Calculus: A Tutorial**
Robin Milner
- LFCS-91-181 : Coherence in Category Theory and the Church-Rosser Property**
C B Jay
- LFCS-91-182 : Fixpoint and Loop Constructions as Colimits**
C B Jay
- LFCS-91-183 : Long Bn Normal Forms and Confluence**
C B Jay
- LFCS-91-184 : An Interleaving Model for Real-Time Systems**
Chen Liang
- LFCS-91-185 : Decidability and Completeness in Real-Time Processes**
Chen Liang
- LFCS-91-186 : Modelling British Rail's Interlocking Logic: Geographic Data Correctness**
M Morley
- LFCS-91-187 : Modelling Reduction in Confluent Categories**
C B Jay
- LFCS-91-188 : An Ideal Model for an Extended Lambda-Calculus with Refinement**
J Agusti et al

⁶Copies of reports and theses may be obtained by writing to Ms Lorraine Edgar at the Departmental address — by email: lme@uk.ac.ed.dcs

- [102] C. P. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science*, volume 2, pages 477–563. Oxford University Press, DEC 1992.
- [103] R. D. Tennent. Denotational semantics. In *Handbook of Logic in Computer Science* (Editors: S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum). Oxford University Press, 1992.
- [104] R. D. Tennent et al. Semantical analysis of specification logic, part 2. *Information and Computation*, 1992.
- [105] P. Thanisch et al. Compact scheme forests in nested normal form. *International Journal of Computer Mathematics*, 45, pages 23–48, 1992.
- [106] P. Thanisch et al. Finding compact scheme forests in nested normal form is NP-hard. *Information and Computation*, 1992.
- [107] N. P. Topham et al. A comparison of two memory models for high performance computers. In *Proceedings of CONPAR/VAPP-92, Lyon*, 1992.
- [108] N. P. Topham et al. Semantics driven computer architecture. In *Proc. Parallel Computing 1991*, page 8. Elsevier Science Publishers, 1992.
- [109] L. Wang. Deriving a correct computer. In *Proceedings of the 1992 International Workshop on Higher Order Logic Theorem Proving and its Applications*, SEP 1992.
- [110] L. Wang. Formal derivation of a computer. In *Correct Hardware Design Methodologies*. Elsevier, 1992.
- [111] A. S. Wight. What does benchmarking mean today? In *8th UK Performance Engineering Workshop*, Sep 1992.
- [112] A. S. Wight et al. Performance evaluation of FDDI using simulation models. In *Proc. IEE Ninth UK Teletraffic Symposium*, page 6. IEE, 1992.
- [113] P. Yeung and D. J. Rees. Resources restricted aggressive scheduling. In *Proceedings of EDAC 92 Conference*, page 10, MAR 1992.

Prof P.J. Scott: *University of Ottawa, Ontario*
Categorical logic, linear logic, proof theory, λ calculus.

Dr G.B. Sorkin: *IBM T.J. Watson Research Center, New York*
Randomised algorithms; quantitative analysis of stochastic optimisation; Monte Carlo simulation.

Prof G.S. Stiles: *Department of Electrical Engineering, Utah State University*
Parallel stochastic optimisation.

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

Dr J Zhou: *Institute of Automation Research, Sichuan, PR China*
Computer networks; distributed systems.

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 is available. In addition, Joint Honours degrees are available in the following disciplines: 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, and 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 speciality. 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.

- [90] T. Reinhardt, M. Mendler, and R. T. Stroup. Die formale validierung einer "bausteintafel" delay-insensitiver grundlemente. In *Proc. ITG/GME/GI Fachtagung Rechnergestützter Entwurf und Architektur mikroelektronischer Systeme, Poster Session*, Darmstadt, NOV 1992.
- [91] G. D. M. Ross. Managing x in a large distributed environment. In *Proceedings of the 4th Annual Conference of the European X User Group*, pages 203–209. European X User Group, SEP 1992.
- [92] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *Proceedings of seventh annual IEEE Symposium on Logic in Computer Science (LICS '92)*, pages 102–109. IEEE Computer Society Press, 1992.
- [93] D. Sangiorgi et al. Barbed bisimulation. In *Proceedings of 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, volume 623, pages 685–695. Springer-Verlag, 1992.
- [94] D. Sangiorgi and A. J. R. G. Milner. The problem of weak bisimulation up to. In *Third International Conference on Concurrency Theory (CONCUR '92)*, volume 630, pages 32–46. Springer-Verlag, 1992.
- [95] D. T. Sannella, S. Sokolowski, and A. Tarlecki. Toward formal development of programs from algebraic specifications: parameterisation revisited. *?*, 29, pages 689–736, DEC 1992.
- [96] D. T. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: model-theoretic foundations. In *Proc. Intl. Colloq. on Automata, Languages and Programming*, page 16. Springer Verlag, 1992.
- [97] D. T. Sannella and L. A. Wallen. A calculus for the construction of modular prolog programs. *Journal of Logic Programming*, 12, pages 147–177, 1992.
- [98] A. K. Simpson et al. Hierarchical meta-logics. In *Proceedings of META-92*, page 13, JUN 1992.
- [99] A. J. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhäuser Boston, DEC 1992.
- [100] A. J. Sinclair. Improved bounds on mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1, pages 351–370, DEC 1992.
- [101] B. U. Steffen, C. B. Jay, and M. Mendler. Compositional characterization of observable program properties. *Theoretical Informatics and Applications*, 26(5), 1992.

- [77] R. J. Pooley. Process algebras and performance modelling. In *Proceedings of the 8th UK Computer and Telecommunications Performance Engineering Workshop*, pages 86–95. BCS Performance Engineering Specialist Group, SEP 1992.
- [78] R. J. Pooley. Review of “performance engineering of software systems” by connie u.smith. *Performance Evaluation Review*, pages 23–24, NOV 1992.
- [79] R. J. Pooley et al. The derivation of functional properties of process based simulation models. In *Proceedings of the EUROSIM Simulation Congress*, page 12. Elsevier, JUN 1992.
- [80] R. J. Pooley and A. L. Song. An editing and checking tool for stochastic petri nets. In *Modelling and Simulation 1992*, pages 228–232. Society for Computer Simulation, Europe, JUN 1992.
- [81] J. Power and C. Wells. A formalism for the specification of essentially-algebraic structures in 2-categories. *International Journal on the Foundations of Computer Science*, 2, pages1–28, 1992.
- [82] R. N. Procter. Groupware — IT’s new wave? *Financial Technology Insight*, pages 8–12, AUG 1992.
- [83] R. N. Procter. Human factors in financial services.IT. *Financial Technology Insight*, pages 8–11, APR 1992.
- [84] R. N. Procter. Information technology in the financial services sector. In *Strategic Change and IT in the Retail Financial Service*, page 22. Prentice Hall, 1992.
- [85] R. N. Procter. Object-oriented software design. *Financial Technology Insight*, pages 9–12, MAY 1992.
- [86] R. N. Procter. Promoting the usability of IT systems. *Financial Technology Insight*, pages 7–8, MAR 1992.
- [87] R. N. Procter et al. HCI: Whose problem is IT anyway? In *Proceedings of the 5th IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, pages 385–396. North-Holland, AUG 1992.
- [88] Y. Rabinovich, A. J. Sinclair, and A. Wigderson. Quadratic dynamical systems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 304–313. IEEE, OCT 1992.
- [89] M. Read and E. Kazmierczak. Formal program development in modular prolog: A case study. In *LOPSTR’91*, page 23. Springer Verlag, 1992.

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, or in one closely related to it. In the final year, each student undertakes a major individual project, and the topics covered are usually closely linked to research work in the Department. Also, final year students in particular are encouraged to attend the research seminars given by both internal and external speakers throughout the year.

Computer Science Courses

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

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

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

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

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

The final degree classification is determined on the basis of performance in examinations, practical work, and project work, over the two final years of study; each year carries equal weight.

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 ³
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

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.

- [65] P. Moscato and M. Norman. A memetic approach for the travelling salesman problem implementation of a computational ecology for combinatorial optimisation on message passing systems. In *Parallel Computing and Transputer Applications*. IOS Press, 1992.
- [66] M. Norman. Bulk synchronous parallelism. *Wotug Newsletter*, 17, Jul 1992.
- [67] M. Norman, P. Thanisch, and K. Chang. Partitioning DAG computations: a cautionary note. In *Parallel Computing: From Theory to Sound Practice* (Eds W. Joosen and E. Milgrom), pages 360-363. IOS Press, 1992.
- [68] O'Hearn and R. D. Tennent. Semantics of local variables. In *Proceedings of the London Mathematical Society Symposium on Applications of Categories in Computer Science, Durham, England, July 20-30, 1991*, volume 177. Cambridge University Press, 1992.
- [69] J. G. Phillips and E. R. S. Candlin. An environment for investigating the effectiveness of process migration strategies on transputer-based machines. In *Transputer Systems - Ongoing Research (Proceedings World Occam and Transputer User Group Technical meeting 15)*, pages 13-23. IOS Press (Amsterdam), APR 1992.
- [70] B. C. Pierce. F-omega-sub user's manual, version 1.2, OCT 1992.
- [71] G. D. Plotkin et al. A logical view of composition. *Theoretical Computer Science*, 1992.
- [72] R. A. Pollack. Typechecking in pure type systems. In *Proceedings LF Workshop, Båstad*, 1992.
- [73] R. J. Pooley. DEMOGRAPHER, a tool for generation of DEMOS simulation and CCS process algebra models from graphical descriptions using x-windows, JUN 1992.
- [74] R. J. Pooley. DEMOGRAPHER PC, a tool for generation of DEMOS simulation and CCS process algebra models from graphical descriptions using VGA graphics, JUN 1992.
- [75] R. J. Pooley. Deriving the functional properties of process based simulation models. In *Proceedings of the EUROSIM '92 Simulation Congress*, pages 229-234. Elsevier, SEP 1992.
- [76] R. J. Pooley. Formalising the description of process based simulation models. In *Proceedings of the 24th Summer Computer Simulation Conference*, pages 93-97. Simulation Councils Inc, JUL 1992.

- [52] E. Kazmierczak. Modularizing the specification of a small database system in extended ML. *Formal Aspects of Computing*, 4(1), pages 42, 1992.
- [53] C. Lengauer and D. Sangiorgi. An improved systolic array for string correction. In *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies*, pages 205–218. Elsevier, 1992.
- [54] C. Lengauer and J. Xue. Specifying control signals for one-dimensional systolic arrays by uniform recurrence equations. In *Proc. Algorithms and Parallel VLSI Architectures II*. Elsevier, 1992.
- [55] Z. Luo. A unifying theory of dependent types: the schematic approach. In *Proc. of Symp. on Logical Foundations of Computer Science (Logic at Tver)*. Springer-Verlag, 1992.
- [56] P. Martin and S. D. Gilmore. Pragmatic experience of the formal specification of a distributed operating system. In *Proceedings of the 5th International Conference on 'Putting into Practice Methods and Tools for Information System Design'*, pages 31–43, SEP 1992.
- [57] M. Mendler and R. T. Stroup. Newtonian arbiters cannot be proven correct. In *Proc. Workshop on Designing Correct Circuits (Eds J. Staunstrup and R. Sharp)*. North-Holland, JAN 1992.
- [58] A. J. R. G. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2, pages 119–141, 1992.
- [59] A. J. R. G. Milner and K. Larsen. A compositional protocol verification using relativized bisimulation. *Information and Computation*, 1, pages 80–108, 1992.
- [60] A. J. R. G. Milner, J. Parrow, and D. J. Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1), pages 1–40, 1992.
- [61] A. J. R. G. Milner, J. Parrow, and D. J. Walker. A calculus of mobile processes, part ii. *Information and Computation*, 100(1), pages 41–77, 1992.
- [62] F. G. Moller, K. Futatsugi, and K. Ohmaki. Proving buffers with OBJ. Technical report, ETL, Japan, 1992.
- [63] F. G. Moller and C. Tofts. Behavioural abstraction in TCCS. In *Proceedings of 19th ICALP*, page 12. Springer-Verlag, JUL 1992.
- [64] F. G. Moller and C. Tofts. An overview of TCCS. In *Proceedings of Euro-micro '92*, page 6, JUN 1992.

Postgraduate Study

The Department offers facilities for postgraduate study by research leading to the degree of PhD and for study towards the degree of MSc in Computer Science. Study for these degrees is normally undertaken on a full-time basis, but it is also possible to study for them on a part-time basis, over longer periods of time.

Degrees by Research

Students studying for the degree of PhD undertake full-time, supervised research for a minimum period of three years. This research will normally be in an area of an existing research specialism within the Department (see *Research Interests*) but proposals in other areas may be considered.

The minimum period of study includes a first year of study in which students are registered as *Supervised Postgraduate Students* prior to registration as PhD candidates. During this first year, students are expected to participate in an appropriate study programme. For students registered for research in the computational theory area, a specific course on advanced topics is provided. Students intending to pursue research into computer systems or other non-theoretical topics may be directed to particular advanced courses and are expected to attend postgraduate study seminars. In order to register as a PhD candidate, students must submit for approval a research proposal which is discussed with a small panel of staff. The degree itself is examined on the basis of a thesis and oral examination⁴. Closing date for PhD applications is 31 March 1993.

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 department⁵. Closing date for MSc applications is 31 March 1993.

⁴Further 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

⁵Further 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

- [39] J. Hillston. A tool to enhance model exploitation. In *Computer Performance Evaluation '92: Proceedings of the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 179–193. Edinburgh University Press, SEP 1992.
- [40] R. N. Ibbett. Parallel computer architectures: Where next? In *Proc. Transputers'92*, pages 364–380. IOS, May 1992.
- [41] R. N. Ibbett and D. J. Rogers. Xbar: a VLSI circuit for bit-sliced packet switching networks. In *Proceedings of IFIP CONGRESS '92*, volume 1, pages 562–570. Elsevier Science Publishers, SEP 1992.
- [42] R. N. Ibbett, N. P. Topham, et al. Editors. *Programming Environments for Parallel Computing*. Elsevier Science Publishers, 1992.
- [43] M. R. Jerrum. Editor. *Combinatorics, Probability and Computing*, 1992.
- [44] M. R. Jerrum. Large cliques elude the metropolis process. *Random Structures and Algorithms*, 3(4), pages 347–359, 1992.
- [45] M. R. Jerrum. Review of “probabilistic analysis of packing and partitioning algorithms” by e.coffman jr and g.lueker. *The Annals of Probability*, 20(4), pages 2164–2167, 1992.
- [46] M. R. Jerrum, B. McKay, and A. J. Sinclair. When is a graphical sequence stable? In *Random Graphs (A.Frieze and T.Luczak, eds)*, volume 2, pages 101–115. Wiley, 1992.
- [47] M. R. Jerrum and U. Vazirani. A mildly exponential approximation algorithm for the permanent. In *Proceedings of the 33rd Annual IEEE Conference on Foundations of Computer Science*, pages 320–326. IEEE Computer Society Press, OCT 1992.
- [48] S. M. Kahrs. Context rewriting. In *Proceedings CTRS'92*, volume 656, pages 21–35. Springer-Verlag, JUL 1992.
- [49] S. M. Kahrs. Polymorphic type checking by interpretation of code. In *Proceedings of the 4th International Workshop on the Parallel Implementation of Functional Languages*, pages 59–70. RWTH Aachen, 1992.
- [50] S. M. Kahrs. Unlimp, uniqueness as a leitmotiv for implementation. In *Proceedings PLILP'92*, volume 631, pages 115–129. Springer-Verlag, AUG 1992.
- [51] K. Kalorkoti. On inverting polynomials and formal power series. *SIAM Journal on Computing*, page 9, 1992.

- [24] M. P. Fourman. The CAD tool of the future. In *IFIP '92 Proceedings of the 12th World Computer Congress Madrid, Spain*, volume 1. Elsevier, 1992.
- [25] M. P. Fourman and W. K. Phoa. A proposed categorical semantics for pure ML. Lncs, Springer-Verlag, 1992.
- [26] S. D. Gilmore. Book review: Action semantics. *University Computing*, 1992.
- [27] S. D. Gilmore. Book review: Programming with VDM. *University Computing*, 1992.
- [28] S. D. Gilmore and M. Clint. Evaluating formal methods. In *Proceedings of the 5th International Conference on 'Putting into Practice Methods and Tools for Information System Design'*, pages 15–29, SEP 1992.
- [29] H. Goguen and Z. Luo. Inductive data types: Well-ordering types revisited. In *Logical Frameworks*, page 19. Cambridge University Press, 1992.
- [30] L. Goldberg. Automating polya theory: The computational complexity of the cycle index polynomial. *Information and Computation*, 1992.
- [31] L. Goldberg. Efficient algorithms for listing unlabeled graphs. *Journal of Algorithms*, 13(1), pages 128–143, 1992.
- [32] K. Goossens. Integrating hardware description languages and proof systems. In *Poster presentation at IFIP '92, Madrid*, Sep 1992.
- [33] K. Goossens. Operational semantics based formal symbolic simulation. In *Higher Order Logic Theorem Proving and Its Applications, Leuven, Belgium*. (Eds: L. Claesen and Michael Gordon), pages 487–506. North Holland, SEP 1992.
- [34] J. Groote and F. G. Moller. Verification of parallel systems via decomposition. In *Proceedings of CONCUR '92*. Springer-Verlag, AUG 1992.
- [35] R. Harper, F. Honsell, and G. D. Plotkin. A framework for defining logics. *Journal of the Association of Computing Machinery*, 1992.
- [36] T. H. Heywood et al. A practical hierarchical model of parallel computation i: The model. *Journal of Parallel and Distributed Computing*, NOV 1992.
- [37] T. H. Heywood et al. A practical hierarchical model of parallel computation ii: Binary tree and ft graph algorithms. *Journal of Parallel and Distributed Computing*, NOV 1992.
- [38] T. H. Heywood et al. Sorting and list ranking on the hierarchical PRAM model. In *4th Symposium on the Frontiers of Massively Parallel Computation*, OCT 1992.

Postgraduate Students 1992-93

Student/Supervisor	Project
Thorsten Altenkirch <i>R.M. Burstall, Z. Luo</i>	Type theoretic program verification; inductive types.
David Aspinall <i>D.T. Sannella, B.C. Pierce</i>	Foundations for computer-assisted proof-checkers.
Maria Cristina Boeres <i>P. Thanisch, G. Brebner</i>	Distributed simulation; parallel processing; distributed algorithms.
Glenn Bruns <i>G.L. Cleland, C.P. Stirling</i>	Formal approaches to safety critical systems design.
Albert Burger <i>P. Thanisch, S.D. Gilmore</i>	Concurrency control and recovery mechanisms in distributed and parallel database systems; object-oriented database systems.
Louise Burnham <i>M.P. Fourman</i>	Formal specification and verification of systems.
Claudio Calvelli <i>K. Mitchell, R. Milner</i>	Relationships between π -calculus and λ -calculus.
Pietro Cenciarelli <i>R.M. Burstall, G.D. Plotkin</i>	Categorical semantics and program logics.
Chen Liang <i>S.O. Anderson, J. Power</i>	Concurrency; process algebras; timed calculi and their models.
Conrad Chin <i>R. Candlin, M.P. Fourman</i>	Distributed ML and the semantics of programming languages.
Søren Christensen <i>C.P. Stirling, F. Moller</i>	Models for distributed systems; formal frameworks for describing concurrent and distributed systems.
Alan Crawford <i>N.P. Topham, R.N. Ibbett</i>	Design of high performance decoupled architectures; compilation techniques for decoupled architectures.

David Crooke <i>R. Candlin</i>	Methods for architecture-independent parallel software development.
Angus Duggan <i>R.J. Pooley, K. Mitchell</i>	Improving code/documentation consistency in literate programming environments
Richard Eyre-Todd <i>R.J. Pooley, D.J. Rees</i>	Tools and techniques for safe data-structure visualisation.
Marcelo Fiore <i>G.D. Plotkin, C.B. Jay</i>	Semantics of programming languages and logics to reason about programs; categorical treatment of non-terminating computations and recursive definitions, and categorical logics for computable functions.
Bo Gao <i>D.J. Rees, R.A. McKenzie</i>	Asynchronous VLSI array architectures for algorithm embedding.
Neil Ghani <i>D.T. Sannella, C.B. Jay</i>	Applications of type theory and category theory to the systematic development of computer programs.
Healdene Goguen <i>R. Burstall, M.P. Fourman</i>	Type theory and specification.
Paul Goldberg <i>M.R. Jerrum, K. Kalorkoti</i>	PAC-learnability of geometrical figures.
Kees Goossens <i>S.O. Anderson, M.P. Fourman</i>	Hardware description languages.
Timothy Harris <i>M.I. Cole, A.J. Sinclair</i>	Practical and theoretical aspects of the shared memory model for parallel computing.
Rycharde Hawkes <i>R.A. McKenzie</i>	Virtual reality.
Claudio Hermida <i>D. Sannella, S. Anderson</i>	Functional programming and program development by transformation.
Roberto Hexsel <i>N.P. Topham, S.O. Anderson</i>	Performance measurement of shared distributed memory multiprocessors with scalable interconnects, that use IEEE's Scalable Coherent Interface (P1596) as a basis.

- [12] R. Burstall and J. Goguen. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1), pages 95–146, JAN 1992.
- [13] R. Burstall and J. McKinna. Deliverable: a categorical approach to program development in type theory. In *Proceedings of the Third Annual BRA Workshop on Types for Proofs and Programs*, MAY 1992.
- [14] E. R. S. Candlin, P. J. Fisk, J. G. Phillips, and N. Skilling. A statistical approach to predicting the performance of concurrent programs. In *Parallel Computing: From Theory to Sound Practice (Proceedings of the European Workshop on Parallel Computing, Barcelona)*, pages 104–107. IOS Press (Amsterdam), MAR 1992.
- [15] N. Carmichael and M. Norman. Parallel processing: the power and the portability. experiments with 'reusable toolkits'. *Future Generations Computer Systems*, 8(1), pages 3–8, JUL 1992.
- [16] L. Chen. An interleaving model for real-time systems. In *Symposium of Logical Foundations of Computer Science - Tver'92*, volume 620, pages 81–92. Springer Verlag, 1992.
- [17] S. Christensen, Y. Hirshfeld, and F. G. Moller. Context-free processes and beyond: The decidability of infinite-state systems. In *Proceedings of ERCIM Workshop*. University of Pisa, DEC 1992.
- [18] S. Christensen, H. Huttel, and C. P. Stirling. Bisimulation equivalence is decidable for all context-free processes. In *Proceedings of CONCUR 92*, volume 458, pages 138–147. Springer-Verlag, AUG 1992.
- [19] G. L. Cleland. ... and the third pig built his house of bricks. In *SERC/DTI Joint Framework for Information Technology Technical Conference*, MAR 1992.
- [20] M. I. Cole. Parallel software paradigms. In *Advances in Parallel Algorithms*, chapter 1, page 25. Blackwell Scientific Publications, 1992.
- [21] M. I. Cole. Writing parallel programs in non-parallel languages. In *Software for Parallel Computers*, pages 315–323. Chapman & Hall, 1992.
- [22] M. F. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. In *Proceedings of the 17th Colloquium on Trees in Algebra and Programming*, volume 581, pages 145–164. Springer-Verlag, FEB 1992.
- [23] M. F. Dam. R-generability, and definability in branching time logics. *Information Processing Letters*, 41, pages 281–287, 1992.

Publications 1992

- [1] M. Abadi, M. Burrows, B. Lampson, and G. D. Plotkin. A calculus for access control in distributed systems. In *Advances in Cryptography - Crypto '91* (Ed: J. Feigenbaum), volume 576, pages 1-23. Springer-Verlag, 1992.
- [2] M. Abadi, L. Cardelli, B. C. Pierce, and D. Rémy. Dynamic typing in polymorphic languages. In *Proceedings of the ACM SIGPLAN Workshop on ML and its Applications*. ACM, JUN 1992.
- [3] P. Anderson. Effective use of local workstation disks in an NFS network. In *LISA VI Conference proceedings*, pages 1-7. Usenix, OCT 1992.
- [4] D. K. Arvind. On the detection of communication-related errors in concurrent programs. *Parallel Computing*, 18, pages 1381-1392, 1992.
- [5] D. K. Arvind and J. Knight. Detection of concurrency-related errors in joyce. In *Second Joint International Conference on Vector and Parallel Processing*, pages 127-132. Springer-Verlag, SEP 1992.
- [6] D. K. Arvind and C. Smart. Hierarchical parallel discrete event simulation in composite ELSA. In *Proceedings of the 6th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation, Newport Beach, USA.*, pages 147-156. Society for Computer Simulation, JAN 1992.
- [7] A. Avron, F. Honsell, I. Mason, and R. A. Pollack. Using typed lambda calculus to implement formal systems on a machine. *Journal of Automated Reasoning*, pages 309-354, 1992.
- [8] D. M. Berry, A. J. R. G. Milner, and D. N. Turner. A semantics for ML concurrency primitives. In *Conference Record of the 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 119-129. ACM Press, JAN 1992.
- [9] J. C. Bradfield. A proof assistant for symbolic model-checking. In *Proceedings CAV '92*, 1992.
- [10] J. C. Bradfield and C. P. Stirling. Local model checking for infinite state spaces. *Theoretical Computer Science*, 96(1), pages 157-174, APR 1992.
- [11] G. Bruns and S. O. Anderson. A case study in the analysis of safety requirements. In *Proceedings Safecomp '92*. IFAC, 1992.

Jane Hillston <i>R.J. Pooley, J.C. Bradfield</i>	Semantics of component-based stochastic models for performance evaluation.
Martin Hofmann <i>D.T. Sannella, T. Stroup</i>	Infinite computations and nondeterminism in type theory.
Anna Hondroudakis <i>R.N. Procter</i>	Parallel systems, computer architecture, graphics and distributed networks.
Tim Hopkins <i>R.N. Ibbett</i>	Local area networks.
Fred Howell <i>R.N. Ibbett, N.P. Topham</i>	Experiments with an object oriented design and simulation environment.
Kayhan Imre <i>R.N. Ibbett, R. Candlin</i>	Performance monitoring and analysis of parallel programs.
Shangjie Jin <i>G. Brebner, R.N. Ibbett</i>	Parallel implementation of OSI communication protocols.
Roope Kaivola <i>C.P. Stirling, J.C. Bradfield</i>	Temporal logics of programs, in particular the modularity aspects of temporal semantics.
Saif Khan <i>K.N.P. Mitchell, S.O. Anderson</i>	Machine assisted proofs of operational semantics.
Thomas Kelly <i>R.N. Ibbett, D.J. Rees</i>	Towards a unified model of parallel architectures.
Lee Yong Woo <i>A.S. Wight</i>	Operating systems, performance modelling and analysis of computer systems.
John Longley <i>M.P. Fourman, G.D. Plotkin</i>	Applications of topos theory to programming language semantics.
Christopher Luth <i>D.T. Sannella, S. Kahrs</i>	Functional programming and algebraic specifications.
Savitri Maharaj <i>S.O. Anderson, Z. Luo</i>	The application of type theory to formal program specification and verification, with a special interest in issues concerning modularity.

Sathiamoorthy Manoharan <i>N.P. Topham, R.N. Ibbett</i>	Object-oriented simulation modelling; performance issues in parallel systems - assignment and scheduling problems.
Paul Martin <i>R. Candlin, S.D. Gilmore</i>	Formal methods in the design of operating systems with hardware monitoring for migration.
Graham Matthews <i>C.P. Stirling</i>	Semantics and language design; program specification, synthesis and verification.
Neil MacDonald <i>S.O. Anderson, M.I. Cole</i>	Predicting and optimising the performance of multicomputer applications.
Simon McPartlin <i>R.A. McKenzie, K. Kalorkoti</i>	Graphical techniques for representing scenes
Michael Mendler <i>R. Burstall, M.P. Fourman</i>	Logics and models for concurrent systems; specification and verification of hardware.
Alexander Mifsud <i>S.O. Anderson, R. Milner</i>	Formal methods of specification and development, especially in CCS.
Matthew Morley <i>R. Milner, G. Cleland</i>	Understanding "safety" in safety critical software and systems through industrially relevant case studies.
Michael Norman <i>P. Thanisch, D. Wallace (Physics)</i>	Parallel computing: the applicability of communication costs models in multicomputers; mapping of parallel computations; the relationship between contention in processor interconnects and scheduling models.
Christopher Owens <i>S.D. Gilmore, S.O. Anderson</i>	Assisted synthesis of implementations of functional programs from formal specifications.
Joseph Phillips <i>R. Candlin, A.S. Wight</i>	The effectiveness of dynamic process allocation strategies within an occam environment.
Randy Pollack <i>R. Burstall, M. Fourman</i>	Computer-assisted proof development, in particular machine-checked mathematics using the Logical Framework and the Calculus of Constructions.

the Science and Engineering Research Council. It also receives substantial support from the University and from its industrial affiliates. It is run under the auspices of an Advisory Board made up of representatives from the funding bodies, the University, industry and users.

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

Other groups within the University, contribute to the work of the Centre and are able to offer a variety of additional facilities to members of the Centre. Membership of the Centre is open to all staff and research students whose work relates to the interests of the Centre. Many members of the Department of Computer Science are also members of the Centre, and a variety of student projects at both undergraduate and postgraduate level involve use of EPCC facilities. The Centre also has a flourishing Summer Student Scholarship Scheme, currently involving some 25 students per annum, in which Computer Science undergraduates frequently participate.

The Edinburgh Parallel Computing Centre

The University of Edinburgh has a long history of involvement in parallel computing and is one of the largest centres of research in the theory and application of parallelism in Europe. Work started in the early 1980s with the use of ICL Distributed Array Processors (DAPs) by the Departments of Physics and Molecular Biology. When a Meiko Computing Surface was acquired in 1986, the Edinburgh Concurrent Supercomputer Project was set up. In 1989, as a way of bringing more closely together the many different aspects of parallel computing research being carried out in the University, the Department of Physics entered into a collaborative venture with the Department of Computer Science and the Edinburgh University Computing Service to establish the Edinburgh Parallel Computing Centre (EPCC). The major goal of EPCC is to ensure that as parallel computers become generally accessible, computational scientists and engineers become able to think naturally of solving their problems on parallel machines, and have the tools and skills they need to do this. To achieve this goal, EPCC supports the applications work being carried out on parallel systems and is investigating those fundamental aspects of parallel computing which can lead to the development of new and improved user support tools.

The Centre is interdisciplinary and acts as a focus for work being carried out within a number of departments. It has subsumed activities on the Edinburgh Concurrent Supercomputer (ECS) and the AMT DAPs, and now provides a service for users on the ECS and DAP facilities (with EUCS support), together with network access to these machines for both local and remote users connected over JANET, and user support relevant to these facilities. Its most recent acquisition is a Connection Machine as part of a collaboration agreement with its manufacturer, the Thinking Machines Corporation of Cambridge, Mass.

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

Organisation and Management

EPCC is supported by major grants from the Department of Trade and Industry, Scottish Enterprise, the Information Systems Committee of the UFC and

Rob Pooley
E.R.S. Candlin, A.S. Wight

A unified framework to support cross paradigm performance modelling.

R. Rangaswami
M.I. Cole, K. Mitchell

Parallelism in functional languages.

Lars Rasmussen
M.R. Jerrum, A.J. Sinclair

Randomisation techniques in enumeration and optimisation.

Vinod Rebello
D.K. Arvind, R.N. Ibbett

A performance analysis of self-timed processor design.

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

Tools and foundations for formal program development.

Davide Sangiorgi
R. Milner, G. Brebner

Semantics of concurrent processes.

Roger Sayle
M.P. Fourman, R. Milner

Use of formal methods, in particular higher order logic in the design of VLSI circuits.

Dilip Sequeira
M.P. Fourman

Semantics and language design.

Peter Sewell
R. Milner

Divergence and abstraction in process algebra.

Alex Simpson
G.D. Plotkin, D. Pym

Type theory and logical frameworks.

Colin Smart
D.K. Arvind

Parallel simulation; distributed algorithms.

Makoto Takeyama
R. Burstall, Z. Luo

Type theory and category theory with a particular interest in fibred categories, applications to proof.

David Turner
R. Milner, K. Mitchell

Functions, modules and concurrency.

Stephen Tweedie <i>E.R.S. Candlin, S.D. Gilmore</i>	Efficient operating systems for parallel programs.
Wang Li-Guo <i>M.P. Fourman, R. Burstall</i>	Proof-based computation synthesis: its theory, methodology and application; formal methods in design of hardware and software.
Amanda Welsh <i>R.J. Pooley, D.A. Welch (Chemistry)</i>	Effective software support for chemical research.
Andrew Wilson <i>K. Mitchell, G.D. Plotkin</i>	Programming language semantics.
Gregory Wilson <i>R.J. Pooley, A.J. Sinclair</i>	Efficient high-level support for parallel programming.

Ph.D. Graduates 1991-92

Simon Ambler
 Simon Brown
 Jorge Farrés-Casaals
 Phillippa Gardner
 James Harland
 Leslie Henderson
 Hans Hüttel
 James McKinna
 Ping Yeung
 Fabio da Silva
 Tom Stiernerling
 Sun Yong
 Jingling Xue

opment using a variety of methods. For example, the Lambda/Dialog system (a product of Abstract Hardware Ltd. – an LFCS affiliate) is used in case studies to investigate high-level design tactics; a delay-insensitive synthesis system based on concurrency theory is being developed; feedback from case-studies is used to direct investigations into appropriate formalisms for this area of specification and verification.

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

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

Industrial Interaction

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

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

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

a state-of-the-art proof assistant which supports the mathematician or computer science to construct formal proofs.

Research in **Concurrency** also focuses on both theories and tools. Increasingly these are being used real in applications (both in LFCS and elsewhere - see below). Theoretical developments are making a number of advances: the development of the π -calculus, a new algebraic model of concurrency which may provide a unified framework for the analysis of concurrent computation; extension of existing process calculi to encompass both timing and probability; the investigation development of modal and temporal logics for concurrent programming; and the use of tableau base proof techniques for reasoning about large or infinite state space system models. Tool development is focused through the concurrency workbench, a rich formal reasoning support environment. Recent developments include a simulation facility and support for timed synchronous calculi.

Investigation into *Language and Semantics* has resulted in the functional programming language Standard ML. It has won wide acclaim and in 1987 was awarded the British Computer Society Award for Technical Achievement. A number of industrially developed and supported SML systems now exist, and it continues to be used as the primary implementation language in LFCS. Work on ML continues in the development of concurrent versions - both in developing a sound theoretical understanding and through practical and efficient implementations on various architectures; and in developing a semantical basis for reasoning about programs written in Standard ML. Other semantically based work includes the investigation of semantical frameworks; the semantics of object oriented programs; and the development of tools and proof strategies for formally reasoning about languages (using the HOL and Lego proof assistants).

A major focus in the area of **Complexity and Algorithms** is the study of randomised algorithms to develop approximate solutions to problems thought to be intractable by analytic means. Recent results in the solution to the Ising problem and in simulated annealing are attracting significant attention. There is also research in the area of algebraic complexity (particularly in bilinear forms) and computer algebra.

The area of **Formal Development of Programs and Systems** contains the widest range of activity, covering theoretical, tool and applications work. Formal program development is focused through the Extended ML system. The basic theory of modular specification and development of programs continues to be investigated and is being supported through tool development and case-studies. Safety critical research has focused on exploitation of several LFCS concepts (including CCS and the concurrency workbench) on a number of industrial case studies, with notable success in several application domains. Future work will be expanded to encompass the problems of formal approaches within "whole" systems development. The formal system design group focuses on system devel-

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) and 60 locally designed and built 68010 ethernet workstations, the "APMs", now used principally for second year undergraduate teaching.

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 $\text{\LaTeX}$ 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.

business meeting, composed of all members of the Laboratory, meet regularly and provide further input to the running of the Laboratory.

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 "themes" within LFCS are outlined below and covered in detail elsewhere but it is worth noting that they unite into a large and coherent research effort. 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 Industry, particularly among those who wish to build upon LFCS research.

The practical part of the research is typified by constructing tools for proving properties of concurrent systems or for constructing and analysing specifications of complex systems. 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 interface mechanisms for constructing proofs or tackling the mathematical problems in integrating a number of logics into a single framework.

Research in LFCS is organised under a number of broad themes. These are, in a sense, artificial as there is considerable overlap between areas and individual research projects will not fit comfortably under a single area. This classification can be useful however in illustrating the broad research profile of LFCS.

Research in the area of **Logics and Proof** is concerned with developing foundational theories with underpin computation, and with developing *effective* computer support for development of proofs in these theories from both a mathematical and computational perspective. Current theoretical interest is in such areas as constructive mathematics, type theories, category theory, linear logic and logical frameworks. Implementation effort is focused through the *lego* proof system,

Laboratory for the Foundations of Computer Science (LFCS)

The groups which now form the Laboratory (LFCS) have a long record of seminal research into the Foundations of Computer Science. This has included both research into the fundamental theory and, in amplification of this, construction of prototype systems and the use of such theory and systems in applications.

This work has exerted a strong influence on the development of Computation Theory particularly in the fields of semantics, specification, 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.

Against this light the Laboratory was formed in January 1986 with the twin aims of intensifying the research and establishing 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 strategic programmes in Informatics and Industry has been awarded to establish the Laboratory and strengthen 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.

LFCS Structure

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

The Laboratory is managed on a day to day basis by its Directorate: Prof. Rod Burstall, Director; Dr. George Cleland, Assistant Director; and Profs. Mike Fourman, Robin Milner and Gordon Plotkin, Co-Directors. An Advisory Board composed of representatives of the rank and file of the Laboratory, internal and external academics and industrialists helps to provide a strategic direction. The research committee, composed of all academic staff members of LFCS, and the

Research Interests

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

Communication Protocols

Research in this area concentrates on the rigorous specification and verification of practical communication protocols. Within the Department, the particular concern is to bridge the gap between work on practical protocols currently in use and work on concurrency (e.g. the Calculus of Communicating Systems (CCS) and Concurrency Workbench described later). Major areas of work are the decomposition of protocols in terms of common communication paradigms, and an investigation of the extent to which computer assistance is possible when verifying protocols. Co-operation with other researchers includes work related to LOTOS, the international standard protocol specification language, which is based on CCS.

Computational Complexity

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.

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.

Computer Architecture and Computer Systems

The Department has been active in computer systems design for many years and has expertise in operating systems, language support and hardware design and implementation, particularly in relation to local area networks, and personal

Inter-Disciplinary Research

Technical knowledge alone is not an adequate foundation for either the design of computer systems, or the evaluation of their performance. Many of the shortcomings of current systems may be attributed to a failure to recognise this fact. Understanding users and their working environment is critical to maximising the usability any computer system. This aspect of system design is addressed by Human Factors. The study of Human Factors is necessarily inter-disciplinary in nature. It covers issues ranging from technologies and techniques for human-computer interaction (HCI), to user cognition, perception and learning, and the context of system use, including job design and organisational factors.

Current areas of Human Factors research interest include:

- The design of group interfaces, and especially the problems of providing effective mechanisms for information sharing and coordination where group members are physically dispersed.
- The design of interfaces for medical applications, particularly in computer-supported radiological diagnosis.
- Interfaces for parallel program development and analysis environments.
- Organisational factors in the design and implementation of IT systems.

Research into organisational factors is undertaken in partnership with members of the University's Research Centre for Social Sciences. One recently completed project examined the problems many organisations face in meeting the growing demand for specialist IT expertise, and integrating it effectively with the more traditional forms of business expertise such as that held by managerial and administrative staff. In depth case studies of IT development practices in firms in the financial services sector were used to explore the difficulties of communication across expertise boundaries.

Very Large Scale Integration

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

Formal Techniques

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

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

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

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.

workstations. At any one time a variety of small projects are undertaken in this general area, together with one or two more major ones. Interest is focusing increasingly on parallel systems and components for constructing them.

The Sparse Project

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

Architectural Simulation

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

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

Concurrent Architectures

Research in this area is centred around the notion of semantically-driven architectures, an implementation technique for parallel architectures in which concepts of action semantics are used to decompose the structure of conventional uniprocessor into three closely coupled and highly specialised sub-processors which

in combination form a high-performance MISD architecture. Active research topics include: semantically-driven architectures with superscalar instruction issue; compilation for semantically-driven machines and memory subsystems for semantically-driven architectures. Tools for investigating these machines are under construction, and include a low-level simulator and compiler.

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. 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.

The Concurrency Workbench

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. Recently a simulation facility has been added. This allows interactive exploration of the system model.

The Workbench has been used to good effect as a teaching aid on courses offered to industrial and academic computer scientists by the LFCS, and has been the topic of seminars for postgraduate students. It has recently been introduced to undergraduate teaching. The analysis and verification facilities have been successfully used to verify a number of classical mutual exclusion algorithms and to analyse communications protocols (one result of these analyses being the discovery of deadlocks in a multi-process implementation of a particular protocol).

project "Mathematically Proven Safety Systems" (MPSS). MPSS has worked with a variety of industrial organisation on analysing and verifying safety properties of systems under development. Sectors in which work has been done includes nuclear protection, railway signalling, and civil aviation.

Work in this area is now moving into a new phase. Through a Teaching Company project we will be working closely with safety consultants Adelard to develop industrial strength techniques for analysis and development of safety-critical systems. The issues of standards, their interaction, the integration of formal approaches to system design, and how all of this can be presented in a safety case which can be objectively assessed by certification authorities, is also being addressed. Correspondents in this work include Shell, British Rail, Lucas, and assessment authorities such as the UK HSE and the German TUV.

Close liaison is kept with the European Workshop on Industrial Computer Systems which in its pre-standards role is well placed to evaluate, exploit and disseminate the results of our work.

Formal and Mechanical Methods of Parallelisation

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

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

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

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

and including intermediate stages in which specification and program are intermingled.

The semantics of Extended ML are based on the primitive specification-building operations of the ASL kernel specification language. From this semantic basis Extended ML inherits complete independence from the logical system (institution) used to write specifications. This allows the use of different logical systems for writing specifications, including logics with special facilities for specifying error conditions, partiality, etc. In principle it also allows any programming language, enriched with ML-style modules, to be adopted for writing code.

Formal Program Development

It is usually easiest to specify programs (in Extended ML or any other specification language) at a relatively abstract level. It is then possible to work gradually and systematically toward a low-level program which satisfies the specification. This will normally involve the introduction of auxiliary functions, particular data representations and so on. This approach to program development is related to the well-known programming discipline of stepwise refinement. The establishment of an appropriate formal notion of the refinement of one specification by another enables this programming methodology to be formalized. If all the refinement steps can be proved correct then the finished program is guaranteed to satisfy the original specification. Since specifications have a modular structure it is possible to develop different parts of the specification independently and assemble the resulting programs later.

Foundational and methodological work has continued into a framework for formal program development. Foundational issues include the formulation of a formal definition of the refinement relation which reflects our programming intuition, proving that refinements compose as required to ensure the correctness of stepwise and modular program development, and generalizing all aspects of the formal program development framework to achieve independence from the logical system used to write specifications. Methodological issues include the use of modular decomposition during the refinement process and the role of interface specifications in the program development process.

Safety Critical Systems

Work in this area aims to employ foundational approaches to the analysis and design of Safety Critical Systems. Such are well suited to formal modelling and analysis. They are usually small, precisely (though informally) specified, and are developed by engineers who are open to any methods which might help to increase confidence in the safety of systems. Up to now this work has focussed through the

A front-end has been constructed which extends the Concurrency Workbench to cover CCS with a restricted form of value passing; this brings a much wider class of applications within its scope. A procedure for minimizing the state space of a process with respect to a modal formula has been implemented. It is also planned to extend the syntax of the process language, which is currently restricted to CCS, by adding operators from other languages, such as CSP and ACP.

Foundational Models

There is a continued strong interest in establishing foundational models, including relating different foundational theories of concurrency such as the algebraic theory of concurrency, net theory, and power domains. Work is progressing on probabilistic versions of Plotkin's power domains. An ESPRIT Basic Research Action seek to unify various concurrency theories, particularly the algebraic ones.

Algebraic Process Calculi

There have been new results extending and elaborating Milner's CCS. A long-standing open problem has been settled (negatively), namely the problem of whether the algebra of finite processes under a certain natural set of operators can be finitely axiomatised. The phenomenon of divergence (infinite internal computation) has been further examined. This work has established a complete algebraic theory (for finite-state processes) for the appropriately refined notion of observation equivalence. CCS has been extended to "mobile processes", i.e. to include the passing of labels as parameters; this has important applications to systems (e.g. operating systems) in which processes migrate among processors. Work on formally relating imperative concurrent while languages to CCS is also being conducted. An emerging major theme in language research is to develop a common theoretical base which fuses functional programming and concurrency. This is also the subject of an Esprit Basic Research Action.

Logics of Concurrency

Hennessy-Milner logic has been extended to include fixpoints. This results in a natural temporal logic for CCS. A new model-checking algorithm has been discovered, and implemented on the Concurrency Workbench. Characteristic formulae for CCS processes have been constructed. General work on defining modal and temporal logics has been undertaken (on relevant modal logics). Interest has also extended to Hoare Logics of concurrent while programs. A study of compositionality in reasoning about concurrent systems is being pursued using these logics.

Languages and Applied Semantics

Standard ML

ML has evolved over a period of about fifteen years. Since 1985 there has been a major effort at Edinburgh to give a precise definition of the syntax and semantics of the language, both of the core language and of the modules language, which is based on David MacQueen's original modules proposal. This work is now completed and in 1987 Robin Milner and Rod Burstall were awarded the BCS Technical award for 'Standard ML'.

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

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

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

A Support Tool for Operational Semantics

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

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

Semantics

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

Formal Development of Programs and Systems

Specifications in ML

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

Performance Modelling

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

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

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

Logics and Proof

Lego: Interactive Proof System for Mathematics and Software Development

We have developed an interactive proof editor, Lego, but based on an extension of Coquand and Huet's Calculus of Constructions. This is a type theory which gives us higher order intuitionist logic. The proof system supports type inference and has both dependent product and dependent sum type. The dependent sum types can be used to represent abstract mathematical structures such as groups and also program inferences. We have been exploring applications to mathematics, for example, theorems about lattices, the construction of real numbers and metric spaces; also about software development, using both categorical ideas and ideas drawn from research on data refinement and algebraic specification. There have been noticeable recent successes in the use of the system for reasoning about Pure Type Systems such as those used in Lego itself.

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.

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.

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.

Computer Graphics

The Department has access to a particularly well-equipped laboratory with a variety of parallel computers of differing architectures within the Edinburgh Parallel Computing Centre (EPCC). These machines can deliver much power to the graphics researcher. Within the EPCC graphics tends to be required to meet the demands of visualisation for large computation applications. In the Department two projects underway both fit into the object-oriented scheme. One line of research involves an investigation of the object-oriented approach to the description and control of all graphical objects and functions in an animated system. The objects themselves, the paths through which they move, the interaction of objects, and so on, and the functions such as illumination objects can all be considered in a similar way. The second project addresses the very real problem of response times in virtual reality systems. The aim is to construct real-time optic and haptic environments for experiments in human perception and control in collaboration with the Department of Psychology here at Edinburgh. The investigation will include examining which features of the optic array are necessary for particular applications and will look at the fundamental principles for the efficacy of virtual reality systems.

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.

Parallelization Tools for Object-Oriented Programming Systems

The provision of general purpose parallelization tools, though desirable, is expected to occur through a generalisation of special purpose tools for particular languages and applications. The work here involves an investigation of the suitability of object-oriented programming languages for parallel computer systems and the design of special purpose tools for use in non-numerical applications. These tools will be restricted in the sense that (a) they are intended for use by an expert programmer who is assumed to have a detailed knowledge of the computational behaviour of the objects; (b) the objects themselves are assumed to be *persistent*, in the sense that their lifetime is typically longer than that of the execution of the applications software; (c) the class hierarchy and the class composition hierarchy are assumed to be relatively stable.

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. A possible implementation medium, springing from the functional characterisation, is to describe each skeleton as a statically distributed graph intended for evaluation by parallel graph reduction. An associated problem is, therefore, to investigate how closely a distributed graph reducer can be made to emulate the dynamic behaviour (in terms of movement and generation of data) of the explicitly parallel algorithm to which the initial graph distribution is intended to correspond. Such experiments are being conducted on a transputer-based Meiko Computing Surface.

The Dynamic Behaviour of Parallel Programs

The POSIE project is directed towards designing operating systems that efficiently support concurrent computations on multiprocessor machines. There are a number of strands to this work.

A hardware test-bed with real-time monitoring facilities has been constructed, and an operating system designed for it. This will permit non-intrusive monitoring, and open up the possibility of effective dynamic load-balancing by process migration. Work is currently being carried out to evaluate the usefulness of such a feedback system in improving performance. The monitor can be made available to an individual user who wishes to tune up a particular program, and the problem of filtering out significant events and presenting information in a helpful way is being investigated. Research is also being carried out to find strategies for automatic run-time reallocation. This ties up with work on the statistical properties of parallel programs and the attempt to find useful quantitative models for predicting program performance.

Simulation is a very useful tool for finding those properties which most affect performance. A number of simple models have been constructed which represent particular types of parallel program, and an investigation is being carried out to see how well they respond to real programs. The aim is to characterise a program in terms of a small parameter set, and to find the most effective allocation strategies for different classes of program. By deriving parameter values at compile-time, or measuring them at run-time, it would then be possible to select an allocation strategy that would have a high probability of leading to good performance.

A simulation tool has been developed which supports the easy modelling of different types of machines and programs. Current work is concerned with evaluating different compile-time mapping heuristics for programs which have a stable behaviour, and with extending the technique to handle run-time reallocation to improve the performance of time-varying programs.