Report on BCTCS 2005

The 21st British Colloquium for Theoretical Computer Science 21-24 March 2005, Nottingham, England

Graham Hutton

The British Colloquium for Theoretical Computer Science (BCTCS) is a forum for researchers in theoretical computer science to meet, present research findings, and discuss developments in the field. It also provides an environment for PhD students to gain experience in presenting their work in a wider context, and benefit from contact with established researchers.

The scope of the colloquium includes all aspects of theoretical computer science, including automata theory, algorithms, complexity theory, semantics, formal methods, concurrency, types, languages and logics. Both computer scientists and mathematicians are welcome to attend, as are non-UK participants.

BCTCS 2005 was held at the University of Nottingham from 21-24 March 2005. The event attracted 80 participants, and featured an interesting and wide ranging programme of 4 invited talks, 2 invited tutorials, and 47 contributed talks. Edited abstracts of all the talks are provided below. Further details, including online copies of the slides from the talks, are available from the BCTCS website at `http://www.bctcs.ac.uk/`. The financial support of the Engineering and Physical Sciences Research Council (EPSRC) and the London Mathematical Society (LMS) is gratefully acknowledged.

BCTCS 2006 will be held at the University of Wales Swansea from 4-7 April 2006. Researchers and PhD students wishing to contribute talks concerning any aspect of theoretical computer science are warmly welcomed to do so. Further details are available from: `http://www.bctcs.ac.uk`.

Invited Talks

Alan Gibbons, King's College London;
Martyn Amos, University of Exeter
The Soft Machines: Computing with the Code of Life

Cellular computing is a new, rapidly expanding field of research at the intersection of biology and computer science. It is becoming clear that, in the 21st century, biology will be characterized more often as an information science. The flood of data generated first by the various genome projects, and now by large-throughput gene expression, has led to increasingly fruitful collaborations between biologists, mathematicians and computer scientists. However, until recently, these collaborations have been largely one-way, with biologists taking mathematical expertise and applying it in their own domain. With the onset of molecular and cellular computing, though, the flow of information has been growing in the reverse direction. Computer scientists are now modifying biological systems to obtain computing devices. Cells are being re-engineered to act as microbial processors, smart sensors, drug delivery systems and many other classes of device. This talk traces the brief history of cellular computing and suggests where its future may lie.

Andrew Gordon, Microsoft Research, Cambridge
Samoa: Formal Tools for Securing Web Services

An XML web service is, to a first approximation, a wide-area RPC service in which requests and responses are encoded in XML as SOAP envelopes, and transported over HTTP. Applications exist on the internet (for programmatic access to search engines and retail), on intranets (for enterprise systems integration), and between intranets (for the e-science Grid and for e-business). Specifications (such as WS-Security) and toolkits (such as Microsoft's WSE product) exist for securing web services by applying cryptographic transforms to SOAP envelopes.

The underlying principles, and indeed the difficulties, of using cryptography to secure RPC protocols have been known for many years, and there has been a sustained and successful effort to devise formal methods for specifying and verifying the security goals of such protocols. One line of work, embodied in the spi calculus of Abadi and Gordon and the applied pi calculus of Abadi and Fournet, has been to represent protocols as symbolic processes, and to apply techniques from the theory of the pi calculus, including equational reasoning, type-checking, and resolution theorem-proving, to attempt to verify security properties such as confidentiality and authenticity, or to uncover bugs.

The goal of the Samoa Project is to exploit these recent theoretical advances in the analysis of security protocols in the practical setting of XML web services. This talk will present some of the outcomes of our research, including: automatic analysis of hand-written pi-calculus descriptions of WS-Security protocols; automatic generation of pi-calculus descriptions from WSE config and policy files; and formal analysis of the WS-Trust and WS-SecureConversation specifications.

Ralf Hinze, University of Bonn
Number Systems and Data Structures

Data structures are a lot like number systems: adding an element to a container corresponds to incrementing a number, deleting an element from a container corresponds to decrementing a number. This talk takes a closer look at the intimate relationship between data structures and number systems. We show, in particular, how number systems with certain properties can be utilised in the design and analysis of data structures. To this end, we discuss various number systems – some well-known, some less so – and show how operations on container types can be modelled after their number-theoretic counterparts. As a worked-out example, we introduce a variant of finger trees that supports deque operations in amortised constant time, and concatenation and splitting in time logarithmic in the size of the smaller piece. Finally, we exhibit the number systems that underly well-known data structures such as red-black trees and 1-2 brother trees.

Rajeev Raman, University of Leicester
Succinctness

There are now good reasons to be re-assessing the space requirements of data structures. This survey will address the following two questions: what space bounds should we be aiming to achieve, and how do we achieve these space bounds and allow efficient operations? We will be using large text indices and XML data as motivating examples.

Invited Tutorials

Roland Backhouse, University of Nottingham

Games for Algorithmic Problem Solving

Combinatorial games provide a very fruitful source of examples of algorithmic problem solving because they are all about winning – identifying an algorithm that will guarantee that you win and your opponent loses! The theory of combinatorial games is also a fruitful source of illustrations of advanced algebraic structures (fields, vector spaces etc.), with the advantage that the problems are concrete and easily understood, but their solution is often very difficult.

This tutorial focuses on using combinatorial games to illustrate important concepts in fixed-point calculus. After a short introduction and overview, we use so-called "loopy games" (games that are not guaranteed to terminate) to illustrate least and greatest fixed points; we also relate properties of winning and losing to calculational rules, like the so-called "rolling rule" of the fixed-point calculus.

Conor McBride, University of Nottingham

Dependently Typed Programming: An Epigram Induction

Types characterize the data and programs which in some way 'make sense'. Dependent types – types expressed in terms of data – can capture notions of 'sense' which are relative to that data. There are many such notions which inhabit our heads but not so typically our programs. Dependently typed programming seeks to reduce this gap between ideas and their execution. For example, we can take $i \times j$ to be the type of matrices with i rows and j columns, then give multiplication the type $\forall i, j, k.\ i \times j \to j \times k \to i \times k$.

Epigram is a dependently typed functional language, equipped with an interactive programming environment. Epigram programs elaborate to constructions in Type Theory by a process strongly resembling inductive theorem proving. This is no idle coincidence. Epigram uses induction princples – expressed by types and interpreted by programs – to characterize derivable notions of pattern matching and recursion in a flexible and extensible way. The language and system continue to be developed at Durham, St Andrews, Royal Holloway and in Nottingham. I shall illustrate Epigram by example, working 'live' at the machine, journeying from induction principles to functional programs and back again.

Contributed Talks

Thorsten Altenkirch, University of Nottingham

Is Constructive Logic Relevant for Computer Science?

Modern Mathematics is based on classical logic and Zermelo-Fraenkel set theory. In this talk I discuss why a constructive approach, such as Martin-Löf's Type Theory, may be more appropriate for Computer Science.

Tony H Bao, University of Wales Swansea

Foundations of a Generic C++ Library
for Generalised Satisfiability Problems

The Satisfiability Problem (SAT) is an actively researched area in computational

theory. While many SAT Solvers were contributed over the years, most of them are written or rewritten from scratch due to the immense implementation details required for increasing efficiency and very often generality was sacrificed for the need for speed. However, with the help of Generative Programming, generality and efficiency can coexist and play nicely together. In this talk I give an overview of work on a C++ library aimed at providing a generic algorithmic framework for solving generalised SAT problems with a focus on both generality and efficiency.

Russell Boyatt, University of Warwick
Modelling and Specification in the Development and Analysis of Communications Protocols

Communications protocols consist of a set of rules to govern the communication between agents. Many such protocols were first developed without a rigorous specification, and formal analysis has typically been first ventured only after problems have emerged in practice. A key problem in applying modelling and specification techniques in this context is identifying suitable abstract properties that can be usefully studied in isolation from the actual physical components of the implementation. We explore this issue and its implications by comparing and contrasting the role that formal specification has so far played in the practical development and analysis of authentication and network protocols.

Paul Bell, University Liverpool
Undecidability of the Membership Problem for a Diagonal Matrix in a Matrix Semigroup

We prove the undecidability of diagonal matrix reachability in a matrix semigroup. Since our proof shows that the above problem is undecidable in dimension 5, it sets a new bound on the problem (currently open in all dimensions). We show a number of encodings to reduce the Post correspondence problem to the reachability of a diagonal matrix in a 5×5 matrix semigroup.

William Blum, University of Oxford
Termination Analysis of Lambda-calculus and a Subset of Core ML

Lee, Jones and Ben-Amram introduced "size-change termination," a decidable property strictly stronger than termination, and proposed the so-called "size-change principle" to analyze it. I propose an extension of this principle to a subset of ML featuring ground type values, higher-order type values and recursively defined functions; this is the first time that this pronciple is applied to a higher-order functional language. The language handles natively if-then-else and let rec structures. The resulting algorithm produces the expected result for higher-order values but can also analyze the size of ground type values. This enhances the scope of the termination analyzer to some recursively defined function operating on numbers.

Mark Callanan, University of Kent
Type-safe Clipboard Operations for Document-Centred Functional Programs

Most functional programming systems consist of a separate editor and compiler/run-time environment, and static type checking is performed after editing. However, with a document-centred system, there is no separation between the editor and the

run-time environment. In this talk we consider combining functional programming with a document-centred interface. The main focus is on the advantages which type-checking brings to the user, and the best ways to provide clipboard operations and other direct manipulation facilities in a type aware environment.

James Chapman, University of Nottingham
Checking Dependently Typed Programs

The goal of this work is to present an independent type checker for Epigram programs. Epigram elaborates programs written in its high-level syntax into the low-level formal language of type theory. Constructions for even modest programs are large and complex; we cannot hope to just look at them to convince ourselves that they are correct. An idea put forward by Pollack is to write a small type checker that we can understand directly to provide ourselves with the necessary conviction that our programs are correct. I report on progress made to this end.

Olaf Chitil, University of Kent
A Theory of Tracing Pure Functional Programs

There exist several tracing systems for Haskell demonstrating the practical viability of different tracing methods. However, combining different methods and generalising them further proves hard to do and many anomalies and defects in existing systems have come to light. To overcome these problems we need simple models of the tracing systems and a theoretical foundation for tracing. In this talk I describe several tracing methods, and a number of problems and shortcomings that have come to light. Finally I outline the semantical theory that we intend to establish to describe both strict and non-strict languages and prove the correctness of various fault location algorithms.

Jolie de Miranda, University of Oxford
The MSO Theory of Level-2 Term Trees is Decidable

Software verification, unlike hardware verification, often gives rise to infinite state systems, so standard techniques successfully employed on hardware rarely carry over to software. This talk unearths various classes of infinite structures with decidable model-checking properties. In particular, we investigate higher-order grammars as generators of infinite trees; such grammars generate infinite term trees and these term trees can be thought of as an accurate model of the syntax trees of higher-order recursive programs. Knapik et al. (2001) showed that, subject to a particular syntactic restriction on the grammars known as "safety," these trees were ripe for model checking, i.e., they possessed a decidable MSO theory. We show that (at least up to level 2) this safety restriction can be removed.

Aleksandar Dimovski, University of Warwick
Game Semantics and CSP Based Approach for Software Model Checking

We present an approach to software model checking based on game semantics and CSP. Open program fragments are compositionally modelled as CSP processes which represent their game semantics. This translation is performed by a prototype compiler. Observational equivalence and verification of properties are checked by trace refinement using the FDR tool. The effectiveness of our approach is evaluated on several examples.

Attila Egri-Nagy, University of Hertfordshire
Algebraic Decompositions of Finite Automata and Formal Models of Understanding

Wreath product decomposition of finite state automata has many potential applications: it could yield automated object-oriented programming in software development; hierarchical decompositions may serve as formal methods for understanding in artificial intelligence; the least number of levels needed for decomposition provides a widely applicable integer-valued complexity measure; and it relates to conservation laws and symmetries in physics. The algebraic theory of such decompositions was first developed in the '60s, but there has never been a computational implementation until recent work carried out under the supervision of Prof Nehaniv. This talk briefly delineates algebraic decomposition theory and reviews the variations of the algorithms presented in different proofs, then discusses results from the first computational implementation.

Osama Elhassan, University of Leicester
Architectural Support for Collaboration Technology for Socio-Technical Systems

The main objective of this proposal is to build a coordination-based solution for supporting modelling of dynamic human-centric collaborative systems. We focus on how modelling predictable human interactions, in terms of norms and norm violation, and externalizing their emerging properties can contribute to managing human components in a flexible way that can respond easily to evolving social and organizational settings as well as new requirements. These properties shall be conceived by studying how to correlate institutional power for modelling norms, category theory for specifying components and their architectural connectors, and temporal logic for injecting time and state notions.

Thomas Erlebach, University of Leicester
Network Discovery and Landmarks in Graphs

Consider the problem of discovering the edges and non-edges of an unknown graph. Initially, only the vertex set of the graph is known. A query at a vertex reveals all edges that lie on shortest paths between that vertex and the rest of the graph. The goal is to minimize the number of queries made until all edges and non-edges have been discovered. We discuss upper and lower bounds on the competitive ratio achievable by on-line algorithms for this problem. We also give results for the off-line version of the problem, which is equivalent to the problem of placing landmarks in graphs.

Neil Ghani, University of Leicester
Coherence via Confluence

Coherence problems are widespread within category theory and, although individual problems have been addressed in detail, there seems to be no systematic way of tackling them. We show how coherence can be studied uniformly by using a 2-dimensional version of Knuth Bendix completion which constructs not only a complete rewrite system, but also an equational theory extending Levy's permutation equivalence. We exemplify our theory with a number of examples.

Andy Gimblett, University of Wales Swansea
Parsing and Static Analysis of CSP-CASL

We formalise the syntax and static semantics of the specification language CSP-CASL in the style of Natural Semantics. We then show how to build a parser and a static analyser according to these definitions within the framework of Hets, the Heterogeneous Tool Set for CASL. On the technical side this uses the combinator parser library Parsec in the functional programming language Haskell. All this takes place within the wider context of developing tool support (theorem prover, model checker, tools for computing different representations) for CSP-CASL.

Jonathan Grattage, University of Nottingham
A Compiler for a Functional Quantum Programming Language

We introduce a compiler for the functional quantum programming language QML developed in Haskell. The compiler takes QML expressions as input and outputs a representation of quantum circuits (via the category FQC of finite quantum computations) which can be simulated by the simulator presented here, or by using a standard simulator for quantum gates. We briefly discuss the structure of the compiler and how the semantic rules are compiled.

Gregory Gutin, Royal Holloway, University of London
Level of Repair Analysis and Minimum Cost Homomorphisms of Graphs

Level of Repair Analysis (LORA) is a prescribed procedure for defence logistics support planning. For a complex engineering system containing perhaps thousands of assemblies, sub-assemblies, components, etc. organized into several levels of indenture and with a number of possible repair decisions, LORA seeks to determine an optimal provision of repair and maintenance facilities to minimize overall life-cycle costs. For a LORA problem with two levels of indenture with three possible repair decisions, which is of interest in UK and US military and which we call LORA-BR, Barros (1998) and Barros and Riley (2001) developed certain branch-and-bound heuristics. The surprising result of this talk is that LORA-BR is, in fact, polynomial-time solvable. To obtain this result, we formulate the general LORA problem as an optimization homomorphism problem on bipartite graphs, and reduce a generalization of LORA-BR, LORA-M, to the maximum weight independent set problem on a bipartite graph.

Will Harwood, University of Wales Swansea
Weak Bisimulation Approximants

Bisimilarity is the canonical behavioural equivalence for process algebras, and weak bisimilarity generalises this by admitting silent actions. Weak bisimulation approximants approach weak bisimilarity by considering games whose length is bounded by an ordinal number; for example, the nth approximant relates exactly those processes whose behaviour cannot be distinguished in a game lasting n turns. On BPP, a simple process algebra, it has long been conjectured that one never need play a game lasting more than $\omega \times 2$ steps; however, currently the only proven bound is the Church-Kleene ordinal (the least non-recursive ordinal number). I show how the bound can be brought down to ω^ω, and suggest an approach for resolving the $\omega \times 2$ conjecture.

Hongmei He, University of Loughborough
Experiments and Optimal Results for Outerplanar Drawings of Graphs
We review our experiments with heuristics and genetic algorithms for low crossing outerplanar drawings and on classes of graphs with optimal outerplanar drawings.

Michael Hoffmann, University of Leicester
Computational Classes of Monoids
In this talk we concentrate on the following four computational classes of monoid: rational, automatic, asynchronously automatic and hyperbolic. We study algebraic and computational properties of the relations between these classes of monoids. We present the complete inclusion structure of these classes of monoids; in doing so we answer open questions concerning the relationship between automatic and hyperbolic monoids: hyperbolic monoids are not necessarily automatic.

Catherine Hope, University of Nottingham
Accurate Step Counting
Starting with an evaluator for a language, an abstract machine can be mechanically derived using successive program transformations. This has relevance to studying both the space and time properties of programs because these can be estimated by counting transitions of the abstract machine and measuring the size of the additional data structures needed, such as environments and stacks. In this talk I use this process to derive a function that accurately counts the number of steps required to evaluate expressions in a simple language.

Matthew Johnson, University of Durham
The Source Location Problem in Digraphs
We are concerned with the problem of selecting the location for facilities, subject to some constraints, in a given network. The nodes selected are called sources, and users at other nodes access the facilities at the sources through the network. Thus a measure of the robustness of the network is the number of disjoint paths between each user and the sources. More formally, a set of (k, l)-sources for a digraph $D = (V, A)$ is a subset $S \subseteq V$ such that for any $v \in V$ there are k arc-disjoint paths that each join a vertex of S to v and l arc-disjoint paths that each join v to S. The Source Location Problem is to find a minimum size set of (k, l)-sources. We provide a polynomial algorithm to solve this problem and discuss progress on other variants of the Source Location Problem.

Mark Jago, University of Nottingham
Belief Revision for Resource Bounded Agents
Rationally updating and revising beliefs is traditionally thought to be possible only for computationally ideal agents that can compute all consequences of their beliefs, regardless of time and space limitations. Focusing on rule-based agents, we show how resource bounded agents can revise and update their beliefs in time linear in the size of the agent's state and program. We argue that such operations are rational, as they satisfy all but one of the AGM postulates for revision. We also discuss the relationship between belief revision and update.

Oliver Kullmann, University of Wales Swansea
Combinatorial Tools for Propositional Satisfiability Decision

Boolean formulas in CNF (conjunctive normal form) are the dominating input format for SAT (satisfiability) solvers. I discuss a natural (and well-known) generalisation of CNF's as "combinatorial data structures," which leads to a very natural approach to (hyper)graph colouring problems; a theoretical application, generalising a theorem of Seymour, is outlined.

Alexander Kurz, University of Leicester
Logics for Transition Systems from Representations of Functors

Coalgebras $X \to TX$ for a functor T on a category C are a well-established model for transition systems. We discuss how specification languages for these transition systems are obtained from the dual functor L of T on the Stone dual of the category C. The dual L itself yields only 'abstract propositions' (meaning equivalence classes of formulas up to interderivability), neither a practical definition of the formulas nor a calculus defining derivability. We introduce the notion of a functor L being presentable by operations and equations and show that each presentation of L gives rise to a sound, complete, and expressive modal logic for T-coalgebras (modal operators are induced by the operations, modal axioms by the equations).

David Manlove, University of Glasgow
Pareto Optimality in House Allocation Problems

An instance of the House Allocation problem (HA) involves a set A of agents, and a set H of houses. Each agent has an acceptable set of houses and ranks this set in strict order of preference. A matching M of agents to acceptable houses is Pareto optimal if there is no other matching M' such that (i) some agent is better off in M' than in M, and (ii) no agent is worse off in M' than in M. I present an efficient algorithm for the problem of finding a maximum Pareto optimal matching, together with related results concerning Pareto optimal matchings. The problem model described here has applications in various contexts, such as the allocation of students to projects, and for the Scottish Executive's Teacher Induction Scheme.

Matus Mihalak, University of Leicester
Joint Base Station Scheduling

We consider the problem where n mobile users want to get data from m base stations. Every base station can send data to any user and this creates an interference around the base station in the form of a disk with radius equal to the distance of the user to the base station. We assume that sending data to a user takes one time unit (one round) and a user can receive data only when it is not in interference produced by another base station. The goal is to assign users to base stations and find the round in which the user is served by the assigned base station while minimizing the number of rounds. We study both one dimensional and two dimensional cases. We present a 2-approximation algorithm for the 1D-case and leave the complexity unsolved. For the 2D case we show the problem is NP-hard and show some lower bounds on some natural greedy algorithms.

Neil Mitchell, University of York
Total Pasta: Static Analysis For Unfailing Pointer Programs

Most errors in computer programs are only found once they are run, which results in critical errors being missed due to inadequate testing. If static analysis is

performed, then the possibility exists for detecting such errors, and correcting them. This helps to improve the quality of the resulting code, increasing reliability. A static analysis has been developed and implemented for the experimental pointer based language Pasta, which is designed to represent pointer manipulating code, such as data structures. The analysis checks for totality, proving that a particular procedure cannot crash and will always terminate. Where a procedure does not satisfy this, the preconditions for the procedure are generated.

Alberto Moraglio, University of Essex
Geometric Interpretation of Crossover

Boiling down the non-algorithmic differences, the various evolutionary algorithms differ only in the solution representation and the relative genetic operators (mutation and crossover) customized for the specific representation. A way to treat different representations uniformly is therefore prerequisite for unification. The focal questions of my research are: i) what is mutation; ii) what is crossover; and iii) what is common to all mutation operators and all crossover operators beyond the specific representation? In this talk I give my answers to these questions and discuss a number of surprising implications.

Peter Morris, University of Nottingham
Generic Programming in a Dependently Typed Language

It is possible using dependent types to write equality functions that give results in a more informative type than just the booleans: we can construct as a result either a proof that the two are in fact the same thing or a proof that they are not. Another dependently typed trick is to use reflection to define the type of regular types (with constructors μ, $+$, $\times$, 1, 0, etc) and then the type of elements for a given regular type (in for μ, inl/inr for $+$, pairing for $\times$, etc). By combining reflection and a more informative equality testing we can write a data-type generic equality test that is obviously correct from its type.

Peter Mosses, University of Wales Swansea
Tool support for Modular SOS

Modular SOS (MSOS) is a variant of conventional SOS. Using MSOS, the transition rules for each construct of a programming language can be given incrementally, and do not need reformulation when further constructs are added to the described language. MSOS thus provides an exceptionally high degree of modularity in semantic descriptions, removing a shortcoming of the original SOS framework. The crucial feature of MSOS is that labels on transitions are now morphisms of a category, and exploited to ensure the appropriate flow of information (such as environments and stores) during computations. The talk first recalls the foundations of MSOS, and illustrates how MSOS descriptions are written in practice. It then introduces a Prolog-based system that can be used for validating MSOS descriptions. An earlier version of the system has been used to support undergraduate courses on semantics at Aarhus and Monash.

Wasana Ngaogate, University of Warwick
Extensible Knowledge Space

This research proposes a model of extensible knowledge and its dynamic change

using classical computer science. The model has been systematically developed into working educational software for module organisation.

Bruno Oliveira, University of Oxford
Exploring Lightweight Implementations of Generics

In "A lightweight implementation of generics and dynamics," Hinze presents a simple approach to generic programming that does not require major extensions to the Haskell 98 type system. Type representations are used to "typecase" over the structure of types. While the approach is quite powerful, supporting generic functions over nearly all Haskell datatypes, it has still some limitations. One of these is the impossibility to override the behaviour of a polytypic definition for some specific instances. I show how to address this problem using ad-hoc polymorphism. This can be readily implemented in Haskell using a two-parameter type class, however this has some drawbacks. Those drawbacks could be removed with a small language extension.

Cristovao Oliveira, University of Leicester
A Framework Based on Coordination and Software Architecture for Mobile Systems

The goal of this work is to develop a methodological approach and support tools for the modelling of distributed and mobile systems along three architectural views: Computation, Coordination and Distribution. This approach has a sound semantic basis over the CommUnity architectural approach developed by Fiadeiro, Lopes and Wermelinger. The views to be developed will also provide support for the distributed execution of the modelled system. Finally, the CommUnity Workbench developed by the candidate in his MSc thesis will be extended to the overall framework, thus providing a tool for modelling mobile systems according to the developed architectural views, following the developed methodology, and animating them using the developed technologies.

Graham Oliver, University of Leicester
Automatic Presentations and Classes of Semigroups

Structures presentable by finite automata are of growing interest, particularly as the closure properties of automata give a generic algorithm for deciding first order properties. The work presented here is a contribution to the foundational problem: given a class of structures X, classify the members of X which have an automatic presentation. We consider various interesting subclasses of the class of finitely generated semigroups. In particular, a classification for the class of finitely generated groups allows a direct comparison with the theory of automatic groups.

Detlef Plump, University of York
Computational Completeness of Rule-Based Languages

We study the computational completeness of rule-based programming languages on arbitrary domains. We only assume a universe of rules and a domain of objects such that rules induce binary relations on the domain. Programs are built from three constructs: nondeterministic application of a finite set of rules, either (a) in one step or (b) as long as possible, and (c) sequential composition of programs. We present a completeness condition guaranteeing that every computable

binary relation (and hence every computable partial function) on the domain is computed by some program. Instantiating the abstract framework with string, term or graph rewriting yields computationally complete languages. In each of these cases, omitting either the construct "as long as possible" or the sequential composition results in an incomplete language. For string and graph rewriting, the one-step application of rules cannot be omitted either and hence the string and graph rewriting languages are minimal.

Markus Roggenbach, University of Wales Swansea
CSP-Prover

We describe CSP-Prover, a theorem prover dedicated to refinement proofs within the process algebra CSP. It aims specifically at proofs on infinite state systems, which may also involve infinite non-determinism. Semantically, CSP-Prover offers both classical approaches to denotational semantics: either based on complete metric spaces or on complete partial orders. Technically, CSP-Prover is based on the generic theorem prover Isabelle, using the logic HOL-Complex. Within this logic, the syntax as well as the semantics of CSP is encoded, i.e., CSP-Prover provides a deep encoding of CSP. Our applications include refinement proofs (1) in the context of an industrial case study on formalising an electronic payment system and (2) for the classical example of the dining mathematicians.

Ana Salagean, University of Loughborough
On the Computation of the Linear Complexity and the k-error Complexity of Binary Sequences With Period a Power of Two

The linear complexity of a sequence is a fundamental parameter for virtually all applications of linearly recurrent sequences, including cryptography. In this talk we show that one can use the Games-Chan, Stamp-Martin and Lauder-Paterson algorithms to compute the analogue notions of linear complexity and k-error linear complexity for finite sequences. Using ideas from the Stamp-Martin and Lauder-Paterson algorithms we develop a new algorithm which computes the minimum number of errors needed to bring the linear complexity of a sequence below a given value. This algorithm has linear (bit) time and space complexity and can be used for encoding and decoding certain binary repeated-root codes. We thus improve on a previous $O(N \log N)$ decoding algorithm of Lauder and Paterson.

Lucy Saunders-Evans, University of Cambridge
Event Structure Semantics for Higher Order Process Calculi

There are many different process calculi for modelling concurrent processes. It is desirable to give a denotational semantics for them that will allow them to be related. Event structure spans appear to provide a very natural semantics for a variety of higher order processes, including Affine HOPLA [Winskel, Nygaard]. However, they are currently limited by the fact that many forms of parallel composition of processes are given an interleaving semantics. I explore some ideas for overcoming this problem and present a small process language together with its semantics with partially synchronous parallel composition as a construct.

Jan Schwinghammer, University of Sussex
A Typed Semantics for Languages with Higher-Order Store and Subtyping

In languages like ML references to functions may be created. The heap store of such languages is referred to as higher-order. While it is fairly straightforward to obtain untyped domain models for languages involving higher-order store, modelling types is harder. Levy presented an elegant typed model of higher-order store based on a possible worlds semantics; I briefly outline its construction before extending the type system with a simple notion of subtyping. A consequence is that derivations of typing judgements are no longer unique. Adapting a proof method due to Reynolds, I prove coherence using a Kripke logical relation and retractions between the typed and untyped models. The resulting semantics is sufficiently expressive to interpret Abadi and Cardelli's (imperative) object calculus.

Sandra Steinert, University of York
Graph Programs for Graph Algorithms

Graph programs as introduced by Habel and Plump provide a simple and yet computationally complete language for computing functions and relations on graphs. We extend this language so that numerical computations on labels can be conveniently expressed. The resulting language GP has a simple syntax and semantics and therefore facilitates reasoning on programs. We present the language and its semantics, and demonstrate its use by giving programs for Dijkstra's shortest path algorithm and showing how to prove correctness and complexity properties.

Chang Su, University of Liverpool
Routing via Single-Source and Multiple-Source Queries in Static Sensor Networks

Sensor networks are a newly emerged and promising field in wireless computing. They consist of a large number of short-range sensor nodes with limited memory, power and computational capacity. These features make communication in sensor networks different from traditional networks. We consider the routing problem in static sensor networks, and show how to finish routing, in both the single- and the multiple-source cases, using limited memory and energy. After preprocessing, we can finish routing in optimal transmissions for the single-source case; for the multiple-source case, it is also an asymptotically optimal algorithm.

Ondrej Sykora, University of Loughborough
The Gap between Crossing Numbers and Outerplanar Crossing Numbers

An outerplanar drawing of an n-vertex graph $G = (V, E)$ is a drawing in which the vertices are placed on the corners of a convex n-gon in the plane and each edge is drawn using one straight line segment. We derive a general lower bound on the number of crossings in any outerplanar drawing of G, using isoperimetric properties of G. The lower bound implies that outerplanar drawings of many planar graphs have at least $O(n \log n)$ crossings. Moreover, for any drawing of G with c crossings in the plane, we construct an outerplanar drawing with at most $O((c+\sum_{v \in V} d_v^2) \log n)$ crossings, where d_v is the degree of v. This upper bound is asymptotically tight. For planar graphs, a outerplanar drawing with the required properties can be constructed in $O(n \log n)$ time.

Rick Thomas, University of Leicester
Groups With A Co-Context Free Word Problem

In this talk we survey some interesting connections between formal language theory on the one hand and group theory on the other. Given a group G generated by a finite set X, the "word problem" of G consists of all words over X that represent the identity element of G. One natural question to ask is about the connections between the complexity of the word problem (as a formal language) and the algebraic structure of G. In this talk we survey some recent results on the set of groups whose word problem is the complement of a context-free language.

Joel Wright, University of Nottingham

Finally, a Simple Semantics

In this talk we develop a simple functional language in which to explore the semantics of synchronous exceptions and asynchronous interrupts. In particular, we seek to define a 'finally' operator in terms of primitives of the language, and prove that this operator has the desired behaviour. This is part of a larger program of work concerned with the semantics of the exception and interrupt handling primitives provided in Concurrent Haskell, and on reasoning about operators and programs written using these primitives.

Xiaohui Zhang, University of Liverpool

New Solution for Multiple Mobile Agent Rendezvous in a Ring

We study the rendezvous search problem for $k \geq 2$ mobile agents in an n node ring. Each agent has only $\log k$ memory, and the one token. Our idea is that we let all the agents start traversing with different speeds, and also that speeds are not fixed. Under some condition, the agent may change its speed. So we can guarantee all of the agents can finally meet together in the non-periodic case.

Paolo Zuliani, Princeton University, USA

Nondeterministic Quantum Programming

In standard computation, nondeterminism provides a way for specifying and reasoning about programs. In quantum computation, nondeterminism is either meant to be "classical" probabilism or it is not considered at all. However, most known examples of quantum computation (e.g. Shor's factoring algorithm and Deutsch-Jozsa's algorithm) have natural nondeterministic specifications. We argue that nondeterminism arises in other examples of quantum computation. In particular, we consider nondeterminism embedded in a programming language for quantum computation, qGCL, and use that to model and reason about quantum nonlocality and quantum mixed-state systems.