Report on BCTCS 2006

The 22nd British Colloquium for Theoretical Computer Science

4–7 April 2006, Swansea, Wales

Faron Moller

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual 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.

BCTCS 2006 was held at Swansea University during 4–7 April 2006. The event attracted 122 participants, and featured an interesting and wide-ranging programme of 6 invited talks and 62 contributed talks; roughly half of the participants and speakers were PhD students. Abstracts for all of the talks are provided below; further details, including on-line 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), the London Mathematical Society (LMS), the British Computer Society (BCS), and the Welsh Development Agency (WDA) is gratefully acknowledged.

A highlight of the meeting this year was a lengthy discussion, led by Samson Abramsky and chaired by Faron Moller, on the formation of a Learned Society for Computer Science in the UK. Such a Society was first proposed for Theoretical Computer Science 18 months earlier in a widely-circulated letter signed by Samson Abramsky, Faron Moller and David Pym which received overwhelming support; this led directly to wider interest in the UK in creating a Learned Society which would envelop the whole of Computer Science. This mammoth endeavour is being developed by a Working Party involving representatives from all relevant national organisations, and the meeting demonstrated near-unanimous support for its efforts (with only a small handful disappointed that the idea of a UK Society just for TCS would now be realised only as a sub-group of a wider organisation).

Another novel feature of BCTCS 2006 was the form of its opening Invited Lecture, which came in the guise of Peter Mosses' Public Inaugural Lecture marking his recent appointment to a professorship at Swansea University.

BCTCS 2007 will be hosted jointly by Oxford and Oxford Brookes Universities in Oxford University's St. Anne's College from 2–5 April 2007. 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 the BCTCS website at `http://www.bctcs.ac.uk`.

Invited Talks at BCTCS 2006

Hajo Broersma, Durham University

Toughness in graphs: structural and algorithmic aspects

(LMS Keynote Lecture in Discrete Mathematics)

The ***toughness*** of a graph G is a vulnerability or reliability measure related to its connectivity, but it involves the number of components $\omega(G-S)$ that result after deleting a subset S of the vertex set of G. Formally, a non-complete graph $G = (V, E)$ is called ***t-tough*** if $t\cdot\omega(G-S) \leq |S|$ for every set $S \subseteq V$ with $\omega(G-S) > 1$. The concept of toughness was introduced by Chvátal in the seventies. Since then a lot of research has been done, mainly relating toughness conditions to the existence of cycle structures. Historically, most of the research was based on a number of conjectures by Chvátal. More recently, research has also focused on computational complexity issues. We will survey progress and open problems in both directions.

Stephen Cook, University of Toronto

A tutorial on proof complexity

NP = co-NP iff there is a propositional proof system in which every tautology has a polynomial size proof. Proving NP ≠ co-NP would show NP ≠ P, and seems out of reach at present. But it motivates trying to prove lower bounds on proof length for specific proof systems. We summarize some results here, and also explain the interesting connection between proof systems and complexity classes.

Tony Hoare, Microsoft Cambridge

Unifying theories of concurrency

The goal of unifying theories is one that inspires much good basic research in all mature branches of scientific endeavour. To show that two or more theories are just special cases of some yet more general theory is a strong support of the credibility of all the theories involved. Unification is a scientific ideal that can justifiably be pursued by theorists for its own sake.

In Computer Science, a good theory of programming can also deliver important practical benefits. It provides a sound conceptual basis for the construction of programming tools, which make the results of the theory available to the practising software engineer. Many such tools, incorporating specialised theories of concurrency like Esterel, are now finding application in the design of hardware and of software systems embedded in aeroplanes and cars.

In order to extend the utility of these tools, it eventually becomes necessary to use them in combination with other specialised tools. To ensure the quality and soundness of such a combined tool, it is essential that it should be based on a unification of their theoretical foundations. So without in any way detracting from the value of proliferation of theories, I would like to suggest that unifying theories is a suitable Grand Challenge to inspire collaborative research in Theoretical Computer Science. I will give a brief example of my own current attempt to unify two familiar theories of concurrency, CCS and CSP. It finds an interesting application of the concept of a Scott retraction.

Mark Jerrum, University of Edinburgh

A tutorial on efficient sampling

Sampling – it might be of points from some spatial distribution or of specified combinatorial structures – is an interesting algorithmic pursuit. Moreover, many sampling problems are externally motivated, for example, by models in statistical physics. There have been a number of notable successes in the area, but many open questions remain. It seems an opportune moment to conduct a SWOT analysis.

Peter Mosses, Swansea University

The meaning of it all: Programming language semantics, from Scott and Strachey to semantics/online

(BCS-FACS Keynote Lecture in Formal Methods)

Since the middle of the last century, hundreds of programming languages have been designed and implemented – and new ones are continually emerging. The texts that make up the programs of a language – the syntax of the programming language – can usually be described quite precisely and efficiently using formal grammars first developed in linguistics. However, the formal description of what the programs do – their semantics – is much more challenging. Like syntax in the 1950s, precise semantics is commonly regarded as impractical and too costly. Research in semantics allows us to reason about software and has provided valuable insight into how (not) to design programming languages, but few semantic descriptions of full languages have been published, and hardly any of these are available online.

One of the major approaches to formal semantics is denotational semantics, developed by Scott and Strachey in the late 1960s. Recent research has shown how to combine some aspects of denotational semantics with other approaches. A radical change to the way semantic descriptions are organised has also dramatically improved their practicality, and should allow efficient online access to a repository of semantic descriptions, as well as contributing to the solution of a long-standing open problem: programming the automatic generation of compilers and interpreters from language descriptions.

Moshe Vardi, Rice University, Houston

Alternation as an algorithmic construct

Alternation was introduced by Chandra, Kozen, and Stockmeyer (JACM, 1981) as a generalization of nondeterminism. The typical use of alternation is in complexity-theoretic settings. The focus of this talk is on presenting alternation as a powerful algorithmic construct. The driving examples are various automated-verification tasks, where alternation yields elegant and optimal algorithms.

Contributed Talks at BCTCS 2006

Thorsten Altenkirch, University of Nottingham
Stop thinking about bottoms when writing programs!

Reasoning about functional programs is often obscured by dealing with the possibility of non-terminating programs, i.e. programs denoting bottom. I argue that most programs can be perfectly well understood in a total setting and this provides a more appropriate framework for reasoning about programs formally and informally. There are some programs where partiality cannot be avoided, eg interpreters, however I will show that this impurity can be dealt with by a monadic interface, the partiality monad. The latter is based on joint work with with Venanzio Capretta and Tarmo Uustalu.

Ioannis Baltopoulos, University of Cambridge
Model checking business processes

In this talk we present ongoing work in the area of model checking of business processes expressed in the pi-calculus. The Business Process Execution Language (BPEL) is an OASIS standard aimed at providing a language for modelling the behaviour of executable business processes and business protocols (abstract processes) collectively known as Business Processes.

Using as a starting point previous work on the semantics of the BPEL language done in Petri nets we initially present a pi-calculus semantics for the language, by means of a set of transformation rules from XML descriptions to pi-calculus ones. The application of the semantics to business processes results in formal process descriptions that lend themselves to property checking through the modal mu-calculus.

Future extensions of this work will involve capturing temporal information in the process descriptions that would enable the calculation of performance metrics and the verification of temporal properties.

Joachim Baran, University of Manchester
Linear temporal logics and grammars

Linear temporal logics are widely associated with automata, i.e. a formula can be (more or less) easily translated into an automaton that accepts the formula's models as its language and vice versa. The most prominent example is probably the linear-time mu-calculus, whose expressiveness coincides with ω-Buchi automata. However, when considering logics that go beyond ω-regular expressiveness, the relationship to automata is either not clear or not straightforward anymore.

The linear-time fixed-point logic with chop (LFLC) is an extension of the linear-time mu-calculus. Its expressiveness is beyond context-free properties and it has been shown that the logic's models coincide with the languages of alternating context-free grammars over finite and infinite words. We give here a short introduction to the field of temporal logics and grammars and our latest research results concerning the representation of the empty word and about the alternation hierarchy in LFLC.

Nick Cameron, Imperial College London

A state abstraction for Java like languages

An objects' state, intended as some abstraction over the value of fields, is always in the mind of (imperative object-oriented languages) programmers. When concurrency comes in, the need for a state abstraction becomes even more prevalent: as the state of an object changes so, usually, does the synchronization behaviour.

We introduce a language feature for expressing the notion of state in Java-like languages. The proposed feature takes the form of a new kind of class, that we call a state class. We first introduce and motivate the new construct through examples written in a dialect of Java called StateJ. Then, we provide a formal account of our proposal by presenting syntax, typing, reduction, and type soundness for the FSJ calculus, a minimal core calculus (in the spirit of Featherweight Java) for modelling the state construct.

Luis Cereceda, London School of Economics

Recolouring graph colourings

Suppose we are given a graph G together with two proper vertex k-colourings of G, α and β. How easily can we decide whether it is possible to transform α into β by recolouring vertices of G one at a time, making sure that at each stage we have a proper k-colouring of G? We consider some algorithmic aspects of this and related questions. Special attention will be given to the case k=3.

This is joint work with Jan van den Heuvel and Matthew Johnson.

David Cunningham, Imperial College London

Implementing atomicity with locks

In a multi-threaded shared-memory system, many single-threaded algorithms fail because of atomicity violations that break the programmer's assumptions. In practice, this is prevented by explicitly coding synchronisation into the program using locking primitives. The programmer attempts to enforce the atomicity that is required for the algorithm to be correct. However this is error-prone, causing hard-to-find bugs in the application. Trying to simplify the solution can result in coarse synchronisation and the loss of the benefits of parallelism.

The atomic section is a much higher level primitive, which programmers can use to directly specify the atomicity requirements of their algorithms. Other threads will never interfere with code in an atomic section. This means there is no possibility of bugs such as deadlocks and race conditions, that can occur when the programmer is forced to implement such a guarantee. The implementation of atomic sections uses transactions to rollback the state if interference occurs, but this is slow without hardware support, cannot support IO in an atomic section, and may have undesirable performance characteristics when threads contend heavily for a resource.

We propose an implementation of atomic sections that uses locks to ensure the atomicity of object oriented code. We define a static analysis on a simple object calculus without dynamic binding. This infers the objects touched by an atomic section, in terms of paths through the initial heap. At run-time these paths are evaluated and the objects "locked" for the duration of the atomic section.

Sharon Curtis, Oxford Brookes University
Multirelational folds

Multirelations are isomorphic to predicate transformers, but provide a different perspective when it comes to the algebraic treatment of angelic and demonic non-determinism. Folds in multirelations take a form very familiar to functional programmers.

Neil Datta, Imperial College London
Computational idioms, symmetry, reversibility and K-theory

Operator algebras have been introduced over the last decade as candidate operational or denotational semantical domains for quantitative extensions of programming languages. Unlike other operator algebras, C*-algebras are topologically stable in infinite dimensions and so it appears they are particularly suitable for the characterisation of recursive behaviour as a limit. Moreover, associated with this topological stability is the property that every C*-algebra is dual to a lattice of projections. There is a general aim to interpret this lattice as an axiomatic semantics which is dual to the operational or denotational perspective encoded in the elements of the C*-algebra.

We are particularly interested in approximately finite-dimensional C*-algebras (AF-algebras). AF-algebras are defined as the limit of an infinite sequence of increasing finite dimensional C*-algebras; each algebra is embedded into its successor by means of an involutive (*-) homomorphism. There is a famous classification theorem for AF-algebras which states that each *-isomorphism equivalence class of AF-algebras is uniquely associated with a "dimension group" (the algebra's "K-theory") arising from its projective structure. We would like to apply this theorem to the long-standing question of comparing the expressiveness of programming languages in a more refined way than merely stating whether or not a language is Turing complete. We suppose that the dimension group somehow characterises the idiom and relative richness of the computational properties or situations that are expressible in a language.

The topological stability of C*-algebras is a consequence of a symmetry constraint arising from the "*" operation. The computational nature of this constraint is not yet clear but is related to reversibility. The behaviour of programs is in general irreversible; it is interesting that naïve AF-encodings of a range of reasonably expressive programming languages produce the same AF-algebra. We will discuss the application of the inherent symmetry of C*-algebras in distinguishing different computational idioms.

Aleksandar Dimovski, University of Warwick
Game semantics supported component verification

Game semantics provides algorithms for software model checking. Open programs are modeled by looking at the way in which they can observably interact with their environment. Computation is seen as a game between two players, the environment and the program, and the model of a program is given as a strategy for the second player.

One of the most important features of game semantics models is compositionality. The model of a large program is constructed from the models of its constituting components (sub-programs). This facilitates using compositional (i.e. divide-and-conquer) verification techniques that decompose the verification of a large program into manageable subparts.

We present a technique for performing compositional verification in an iterative and fully automated fashion; the approach uses learning and model checking.

Mike Dodds, University of York

Graph transformation in constant time

Graph transformation systems are applicable to a wide range of problems including pointer safety, pattern recognition, and forming the basis of graph programming languages. They have the disadvantage, however, that the application of a rule generally requires the construction of a subgraph isomorphism, a problem known to be NP-complete (or polynomial if rules are considered as fixed). In this talk I will discuss so-called "rooted" graph transformation rules, a restricted form of rules where applicability can be checked in constant time (assuming fixed rules). Rooted rules define a set of root vertex labels which can appear at most once in the graph to be transformed, and require that all matched nodes are reachable from some root. I will show that, despite these restrictions, rooted rules are surprisingly general. I will discuss using rooted rules to simulate unrooted rules, and show that rooted rules are general enough for Turing machine simulation. I will also discuss possible applications of rooted rules, in particular in modelling pointer manipulations.

This is joint work with Detlef Plump.

Alastair Donaldson, University of Glasgow

General techniques for symmetry reduction in model checking

Model checking is a potentially useful technique for verifying designs of concurrent systems, but is limited by the state-explosion problem: as the number of components in a concurrent system grows, the state-space of a model of the system suffers combinatorial explosion. Replication in the system being modeled may induce symmetries on the global state-space of a model, and if this replication can be identified in advance, model checking can be performed over a, generally smaller, quotient state-space, consisting of one state per equivalence class with respect to the symmetry. The success of symmetry reduction depends on efficient techniques to compute orbit representatives, and the problem of representative computation is known to be NP-hard.

In this talk, we present exact, efficient solutions to this problem for certain classes of symmetry group, and approximate solutions for arbitrary symmetry groups. We describe an approach to classifying an arbitrary symmetry group based on its structure as a disjoint or wreath product of subgroups, so that an appropriate symmetry reduction strategy can be chosen for the group. We briefly describe TopSPIN, a new symmetry reduction package for the SPIN model checker which implements our techniques.

This is joint work with Alice Miller.

Martin Dyer, University of Leeds

The complexity of counting homomorphisms to directed acyclic graphs

For a fixed (di)graph H chosen from some class, we consider the complexity of the problem of counting the number of homomorphisms to it from some graph G taken from some, possibly different, class. The usual aim is to prove a "dichotomy theorem" showing that for some H the problem is in P and for every other H the problem is #P-complete. We review previous work on this problem, then describe a new result for the problem of counting homomorphisms to directed acyclic graphs, providing a graph-theoretic classification of the "easy" graphs. An interesting feature of the classification, which is absent from previous dichotomy results, is that there is a rich supply of tractable graphs with complex structure.

Edith Elkind, University of Warwick

Nash equilibria in graphical games on trees, revisited

Graphical games have been proposed as a game-theoretic model of large-scale distributed networks of non-cooperative agents. When the number of players is large, and the underlying graph has low degree, they provide a concise way to represent the players' payoffs. It has recently been shown that the problem of finding Nash equilibria on a general degree-3 graphical game is complete for the complexity class PPAD, indicating that it is unlikely that there is any polynomial-time algorithm for this problem. We show here that in contrast, degree-2 graphical games are tractable.

Our algorithm uses a dynamic programming approach, which was introduced by Kearns, Littman and Singh in the context of graphical games on trees. The algorithm of Kearns et al. is a generic algorithm which can be used to compute all Nash equilibria. The running time is exponential, though approximate equilibria can be computed efficiently. Kearns et al. proposed a modification to the generic algorithm in order to find a Nash equilibrium in polynomial time, provided that the underlying graph is a bounded-degree tree. We show that this modified algorithm is incorrect: the output is not always a Nash equilibrium.

In this talk, we focus on graphical games in which the underlying graph is a path or "path-like". First, we show that Nash equilibria can be computed in quadratic time if the underlying graph is a path, and therefore in polynomial time if the underlying graph has maximum degree 2. Our algorithm, which is based on the approach of Kearns et al., can be used to compute Nash equilibria of graphical games on arbitrary trees, but the running time can be exponential, even when the tree has bounded degree. We show that this is inevitable: any two-pass algorithm of this type will take exponential time, even on bounded-degree trees with pathwidth 2.

It is an open question whether our algorithm runs in polynomial time on graphs with pathwidth 1 but we show that finding a Nash equilibrium for a graphical game in which the underlying graph has maximum degree 3 and constant pathwidth is PPAD-complete (and hence is unlikely to be tractable).

This is joint work with Leslie Ann Goldberg and Paul Goldberg.

Simon Foster, University of Sheffield

A formal model for web-service composition

Orchestration describes the rules which define the patterns underlying the way in which a composite web-service interacts with other services, thereby enabling composite functionality. In this talk we look at Cashew-S, a language based on OWL-S, which we have created for specifying orchestration with formal semantics provided by a timed process calculus, and examine how this fits into the general theory of service composition via processes. We believe that this research will eventually enable us to give a formal model for choreography, the method by which services engage in conversations, and will further aid in automated service composition.

Stephen Gorton, University of Leicester

Task-oriented business requirements elicitation for web services

The expression of semantically-rich business requirements for web services is restricted by current composition and management solutions available, e.g. BPEL. These solutions often address orchestration concerns, rather than actual requirements. Methods such as BPMN, although able to express requirements in terms of business activities and flows, are unable to express non-core requirements such as resource and performance constraints. In this talk, we will present a language to accurately express business requirements specifications through the use of graphical notation and policies. We will present a business goal G, which defines a set of tasks T, a task map m and a set of operators O. Subsequently, we will explain how this method can map to current composition and management solutions such as graphical notations and BPEL.

Alexey Gotsman, University of Cambridge

On proving liveness properties of programs

We describe a new counterexample-guided refinement-based algorithm for proving liveness properties of programs built upon an extension of a recently discovered technique for proving program termination. Our implementation of the algorithm provides an automatic, interprocedural, path sensitive and context-sensitive liveness prover for the C programming language. It supports such language features as arbitrarily nested loops, arbitrarily nested recursive functions, pointers and side-effects, and function-pointers.

This is joint work with Byron Cook, Andreas Podelski, and Andrey Rybalchenko.

Jonathan Grattage, University of Nottingham

QML: A functional quantum programming language

QML is a high-level functional quantum programming language that allows the contraction of quantum variables, in apparent contradiction of the no-cloning theorem of quantum mechanics. Of course, no cloning actually takes place. The contraction, or non-linear use of variables, is modeled as sharing, as with most programming languages. This is achieved by the use of two if-then-else constructs: a classical-if, which measures quantum data; and a quantum-if, which allows quantum parallelism and entanglement. We show how these constructs allow contraction-as-sharing, and briefly discuss the issues of orthogonality with regard to judgments needed for the quantum-if.

Alexander Green, University of Nottingham

Reversible quantum circuits from irreversible functions

Quantum circuits are, by their nature, reversible. It is possible to create reversible circuits that perform irreversible functions. These transpositions often require additional input qubits ("heap"), and produce extra output ("garbage"), that allows the function to be reversible. In this talk we give three laws governing circuits containing heap and garbage, show how they can be optimised, and show how they can be used to prove the measurement postulate. The three laws also hold for classical reversible circuits, but other laws are also satisfied such as that the measurement of a bit has no effect on other bits in the circuit.

Abubakar Hassan, King's College London

Compiling interaction nets

Interaction nets have been put forward as both a graphical programming paradigm and as an intermediate language into which we can compile other languages. Although we can already program in interaction nets, they still lack what modern programming languages should offer. In this talk, we address two issues. Firstly, we consider how to expand this programming paradigm to become a useful and usable programming language, providing features such as a module system, data types, input/output etc. Secondly, we consider how to compile such a language by giving a compilation scheme of interaction nets to bcode (our target/intermediate language) which can be executed by our abstract machine, or further be translated into Java bytecodes for execution by a Java virtual machine. We conclude the talk by giving a formal correctness proof of the compiler.

Michaela Heyer, University College Cork

An intelligent example-generator for graph theory

As with most areas of mathematics, graph theory relies heavily on the use of examples and counterexamples to prove, disprove or support new conjectures. The intelligent example-generator has a way of providing these example graphs in a very efficient manner by combining different aspects of computer science and mathematics. In this talk I give a broad overview of the areas involved and describe the main steps in the process of generating the example graphs: the generation of all graphs of a given size; the use of automated logical inference together with expert knowledge of graph theory to greatly reduce the amount of work to be done; and the use of a 100 node Beowulf cluster to speed up the process through parallel testing of graph properties.

Catherine Hope, University of Nottingham

Fusion in less space

There are many advantages to writing functional programs in a compositional style, such as clarity and modularity. However, the resulting programs can be inefficient in terms of space due to the use of intermediate data structures. These structures may be removed using deforestation techniques, but whether the space performance is actually improved depends on the structures being consumed in the same order that they are produced. In this talk I explore this problem and present a solution.

Wan Huang, London School of Economics

Enumerating Nash equilibria for game trees

We develop an algorithm for finding all Nash equilibria in a game tree with imperfect information. The algorithm is based on the "sequence form" which was introduced by von Stengel, and employs linear programming duality and polyhedral theory. The sequence form represents "mixed strategies" in game trees compactly, with exponentially fewer variables than the strategic form. The Nash equilibria are the solutions to an optimization problem (called a linear complementarity problem) derived from the sequence form. We show how to remove all the redundant variables and some of the redundant constraints of this optimization problem by considering terminal sequences of moves. We also give a proof that all Nash equilibria correspond to some convex combination of certain vertices in the polyhedra.

Andrew Hughes, University of Sheffield

Combining timing, localities and migration in a process calculus

Process calculi provide an abstract representation of concurrency, and have been used in both theoretical and practical contexts. Since the early days of CCS, CSP and ACP, process algebras have diverged into two groups: those that add the concept of time, and those that bring in mobility. The latter has been represented in two different ways: as scope mobility (most notably in the pi-calculus) and as migration.

My research attempts to combine these two separate ideas within a single process calculus. This continues my earlier work with the CaSE and Cashew Nuts calculi developed at Sheffield. These both have a notion of time, represented by clock hierarchies, but do not include mobility. I will discuss these briefly, before demonstrating the use of other calculi which allow the representation of distribution, via locality, and process migration. I will also consider how these ideas might be combined.

Markus Jalsenius, University of Warwick

Improved mixing bounds for the anti-ferromagnetic Potts model on Z^2

We consider the anti-ferromagnetic Potts model on the integer lattice Z^2. The model is used in statistical physics and corresponds to graph colourings with two parameters, q and λ: parameter q is the number of colours and $\lambda \in [0,1]$ is used to weight colourings. We are interested in sampling from the set of q-colourings on Z^2 where each colouring is assigned a probability proportional to its weight. In the case $\lambda = 0$ we are sampling from the uniform distribution on proper q-colourings. If $\lambda = 1$ we are sampling from the uniform distribution on all q-colourings. We use Markov chains, Glauber dynamics, to sample colourings. Each state corresponds to a colouring and the stationary distribution is identical to the distribution we want to sample from. It is known that Glauber dynamics is rapidly mixing if $q > 7$, $\lambda \in [0,1]$, or if $q = 7$, $\lambda = 0$, $\lambda > 1/8$, or if $q = 6$, $\lambda = 0$, $\lambda > 1/4$. We show that Glauber Dynamics is rapidly mixing for $q \geq 6$ and any $\lambda \in [0,1]$. We also show rapid mixing for a larger range of λ than was previously known for $q = 3, 4$ and 5.

This is joint work with Leslie Ann Goldberg, Russell Martin and Mike Paterson.

Mauro Jaskelioff, University of Nottingham

Towards operations on operational semantics

Standard structural operational semantics has poor modularity. We build upon the work of Turi's functorial operational semantics, an abstract category-theoretical model of operational semantics which guarantees that the defined semantics are well behaved. Working in this abstract setting we reason about the combination of operational semantics and we define operations for achieving it. We show how to use these operations to combine four toy languages featuring different effects.

Kenneth Johnson, Swansea University

The theory of spatial data types and constructive volume geometry

Spatial data types model data in space. There are a vast number of examples in computing, ranging from medical images to computer memories. Spatial data types are modeled using algebras of total functions from a topological space of points to a topological algebra of attributes.

In this talk, we motivate a general theory of spatial data types by introducing Constructive Volume Geometry (CVG), an algebraic framework for the high-level programming of spatial objects in graphics. We discuss the problem of the expressive power of CVG in the context of the 4-colour channel model: Do the CVG terms define all the spatial objects one wants?

We sketch some general theory and show how to solve the expressiveness problem in general. Using variants of the Stone-Weierstrass Theorem we prove a density result which shows that some subsets of spatial objects under certain primitive operations can approximate all other spatial objects.

Oliver Kullmann, Swansea University

Using (hyper)graph decomposition for satisfiability decision

Given a hard satisfiability problem F (boolean or not), an interesting option is to use some (hyper)graph representation $G(F)$ with the property that, if $G(F)$ splits into connected components, then we can work on the components independently. In this talk, I present a unifying approach, looking at the whole picture from "local" hypergraph cuts to "global" tree decompositions (typically associated with fixed parameter tractability). In this way the different decomposition strategies can be much better linked to the "logical" properties of the problem F (as opposed to the "graphical" properties represented by $G(F)$); we identify the missing link here as the cause for the (hitherto) failure of graph decomposition algorithms for practical applications (while being attractive in theory).

Ranko Lazic, University of Warwick

LTL *with the freeze quantifier and register automata*

Temporal logics, first-order logics, and automata over data words have recently attracted considerable attention. A data word is a word over a finite alphabet, together with a piece of data (an element of an infinite domain) at each position. Examples include timed words and XML documents. To refer to the data, temporal logics are extended with the freeze quantifier, first-order logics with predicates over the data domain, and automata with registers or pebbles.

We investigate relative expressiveness and complexity of standard decision problems for LTL with the freeze quantifier, 2-variable first-order logic (FO^2) over data words, and register automata. The only predicate available on data is equality. Previously undiscovered connections among these formalisms, and to counter automata with incrementation errors, enable us to answer several questions left open in recent literature.

We show that the future-time fragment of LTL with freeze which, corresponds to FO^2 over finite data words, can be extended considerably while preserving decidability, but at the expense of non-primitive recursive complexity, and that most further extensions are undecidable. We also prove that, surprisingly, over infinite data words, LTL with freeze and without the "until" operator, as well as nonuniversality of one-way nondeterministic register automata, are undecidable even when there is only 1 register.

This is joint work with Stephane Demri (CNRS & ENS Cachan & INRIA Futurs).

Cindy Li, University of Liverpool

Efficient probe selection in microarray design

DNA microarray technology, originally developed to measure the level of gene expression, is becoming one of the most widely used tools in genomic study. Microarrays have been proved to benefit areas including gene discovery, disease diagnosis, and multi-virus discovery. The crux of microarray design lies in how to select a unique probe that distinguishes a given genomic sequence from other sequences. However, in cases that the existence of a unique probe is unlikely, e.g. in the context of a large family of closely homologous genes, the use of a limited number of non-unique probes is still desirable.

Due to its significance, probe selection attracts a lot of attention. Various probe selection algorithms have been developed in recent years. Good probe selection algorithms should produce as small a number of candidate probes as possible. Efficiency is also crucial because the data involved is usually huge. Most existing algorithms usually select probes by filtering, which is usually not selective enough and quite a large number of probes are returned. We propose a new direction to tackle the problem and give an efficient algorithm to select (randomly) a small set of probes and demonstrate that such a small set of probes is sufficient to distinguish each sequence from all the other sequences. Based on the algorithm, we have developed a probe selection software RandPS, which runs efficiently and effectively in practice. A number of experiments have been carried out and the results will be discussed.

This is joint work with Leszek Gasieniec, Paul Sant and Prudence WH Wong.

Olga Lightfoot, Queen Mary, University of London

Real arithmetic test suite for a theorem prover

Theorem proving has been used with great success in system verification, in particular, in reasoning about problems expressed in terms of continuous mathematics over the real numbers. Such mathematical expressions lie in the domain of higher order logic, so we are interested in the performance of theorem provers in handling operations on functions in the domain of real numbers. Despite the good level of automation of linear arithmetic operations, non-linear operations over both algebraic and transcendental functions still present a considerable challenge. We discuss the construction of a test suite for evaluating the real arithmetic capabilities of an interactive theorem prover and, using PVS, show what can be deduced about a theorem prover from such a test.

David Manlove, University of Glasgow

Vertex and edge covers with clustering properties: complexity and algorithms

We consider the concepts of a t-total vertex cover and a t-total edge cover ($t \geq 1$), which generalize the notions of a vertex cover and an edge cover, respectively. A t-total vertex (respectively edge) cover of a connected graph G is a vertex (edge) cover S of G such that each connected component of the subgraph of G induced by S has at least t vertices (edges). These definitions are motivated by combining the concepts of clustering and covering in graphs. Moreover they yield a spectrum of parameters that essentially range from a vertex cover to a connected vertex cover (in the vertex case) and from an edge cover to a spanning tree (in the edge case). For various values of t, we present NP-completeness and approximability results (both upper and lower bounds) and FPT algorithms for problems concerned with finding the minimum size of a t-total vertex cover, t-total edge cover and connected vertex cover, in particular improving on a previous FPT algorithm for the latter problem.

This is joint work with Henning Fernau (University of Trier).

Erik Arne Mathiesen, Queen Mary, University of London

Abstract Hoare logic and dynamical systems

Setting out to define a Hoare logic for dynamical systems, we define an abstraction of Hoare logic in the setting of traced monoidal categories. More particularly, we define a class of subcategories of the category of posets and monotone mappings on which we define a sound and complete system of inference rules. This provides us with a way of generating Hoare-logic-like rules for general systems through embeddings into the beforementioned class of subcategories. A particular instance is the embedding of the traced monoidal category of while programs which yields Hoare's original logic. Of special interest to us are embeddings of certain dynamical systems. We conclude by discussing further aspects of the framework such as partial vs total correctness.

Neil Mitchell, University of York

CATCH - Case And termination check for Haskell

There are two main ways in which a Haskell program may fail at runtime. Firstly, the program may not terminate. Secondly, if the program has any incomplete (non-exhaustive) patterns in definitions or case alternatives then it may encounter a pattern match error.

This work presents an automated analysis which checks a Haskell program, and attempts to generate a proof that the Haskell program encodes a total function. It includes a constraint language that can be used to reason about the data in a Haskell program, along with mechanisms to propagate these constraints between program components.

Matthias Mnich, London School of Economics

Computation of correlated equilibria in succinctly-representable games

A correlated equilibrium is a more general notion of a Nash equilibrium modelling the simplest form of co-operation in a game via "shared randomness". The question of whether a polynomial-time algorithm exists for computing correlated equilibria in succinctly representable games has recently been solved by Papadimitriou (STOC 2005). This approach applies certain properties of the ellipsoid algorithm for linear programming that allows it to run on a problem with polynomially-many unknowns and exponentially many constraints. We give an overview of that idea and also include other results encompassing other classes of games. Finally, possibilities to use this algorithm for game trees with imperfect information (called extensive games) are discussed, where the number of pure strategies may be exponential in the size of the extensive game.

Peter Morris, University of Nottingham

Containing families

We examine the idea of indexed-containers from a programming perspective. We show how to use this notion to characterise inductive families in the dependently-typed functional language Epigram. This gives us a flexible and compositional semantic notion of Strict Positivity and enables us to write generic programs not in a type system we model but for the full Epigram type system.

Dimitrios Mostrous, Imperial College London

Session types for object-oriented languages

A session takes place between two parties; after establishing a connection, each party interleaves local computations with communications (sending or receiving) with the other party. Session types characterise such behaviour in terms of the types of values communicated and the shape of protocols, and have been developed for the pi-calculus, CORBA interfaces, and functional languages. We study the incorporation of session types into object-oriented languages through the language Moose, a multi-threaded language with session types, thread spawning, iterative and higher-order sessions. Our design aims to consistently integrate the object-oriented programming style and sessions, and to be able to treat various case studies from the literature.

We describe the design of Moose, its syntax, operational semantics and type system, and develop a type inference system. After proving subject reduction, we establish the progress property: once a communication has been established, well-typed programs will never starve at communication points.

This is joint work with Mariangiola Dezani (University of Turin) and Nobuko Yoshida and Sophia Drossopoulou (Imperial College).

Jonty Needham, University of Bath

A fully abstract game semantics for answer set programming

We present a fully abstract interaction model of answer set programming based on logic games semantics. The field of games semantics has proved to be a powerful mathematical framework in which to view a wide variety of programming languages and logics. We present an outline of games semantics and then prove the correctness result for the denotation. We also present a result about providing an alternative algorithm for grounding which reduces computation time, as well as the correctness of a debugging algorithm.

Gregg O'Malley, University of Glasgow

Stable matching problems with constant length preference lists

The Stable Marriage problem (SM) and many of its variants have been studied extensively in the literature. In recent years much research has focused on the model SMTI, where a given participant's preference list may involve ties and be incomplete. It is currently known that, in this setting, stable matchings may have different sizes, and the problem of finding a maximum stable matching is NP-hard. In this talk we consider various restrictions of SMTI where some of the preference lists are constrained to be of constant length. Such restrictions arise naturally in practical applications. We prove that in some cases, finding a maximum stable matching can be achieved in polynomial time, and for others we show that the problem remains NP-hard.

Nick Palmer, University of Warwick

PAC-learnability of probabilistic deterministic finite state automata in terms of variation distance

We consider the problem of PAC-learning distributions over strings, represented by probabilistic deterministic finite automata (PDFAs). PDFAs are a probabilistic model for the generation of strings of symbols, that have been used in the context of speech and handwriting recognition and bioinformatics. Recent work on learning PDFAs from random examples has used KL-divergence as the error measure; here we use variation distance. We build on recent work by Clark and Thollard, and show that the use of variation distance allows simplifications to be made to the algorithms, and also a strengthening of the results; in particular that using variation distance, we obtain polynomial sample size bounds that are independent of the expected length of strings.

Nick Papanikolaou, University of Warwick
A framework for automated verification of quantum cryptographic protocols

This talk will consider the problem of proving correctness and security properties for communication protocols, and cryptographic protocols in particular, with a view to showing how formal methods may be used in the analysis of schemes for quantum communication. Quantum cryptographic techniques rely on the laws of quantum mechanics to establish a secret key between two users; indeed, several protocols implementing these techniques have been shown to be perfectly secure in the information–theoretic sense.

We will give an account of the process algebra CQP (Communicating Quantum Processes) and how it may be used to build accurate models of quantum protocols; also, the verification of protocol models using a probabilistic model-checker will be discussed. Finally, we will report on our progress in developing an integrated framework for simulating and verifying properties of systems with finite-dimensional quantum state spaces.

This is joint work with Rajagopal Nagarajan (University of Warwick) and Simon Gay (University of Glasgow).

Mike Paterson, University of Warwick
Overhang

How big an overhang beyond the edge of the table can we reach by stacking n identical blocks of length 1? The classical solution achieves an overhang of $1/2H_n$, where $H_n \approx \ln n$ is the n^{th} harmonic number. This solution is widely believed to be optimal. We show that it is exponentially far from optimal.

This is joint with Uri Zwick (Tel Aviv University).

Daniel Paulusma, Durham University
Locally constrained graph homomorphisms and degree refinement matrices

We consider three types of locally constrained graph homomorphisms: bijective, injective and surjective. Degree refinement matrices have tight connections to locally constrained graph homomorphisms. If a graph G has a homomorphism of a given type to a graph H then we say that the degree refinement matrix drm(G) of G is smaller than drm(H). In this way we obtain three partial orders. Computing drm(G) is easy, so an algorithm deciding comparability of two matrices in one of these partial orders would be a heuristic for deciding if G has a homomorphism of a given type to H. For the locally bijective constraint this corresponds to a well-known heuristic for graph isomorphism. For local surjectivity and injectivity we show that the problem of matrix comparability belongs to the complexity class NP.

This is joint work with Jiří Fiala and Jan Arne Telle.

Kasper Pedersen, University of Warwick

A scan Markov chain for sampling colourings

Consider a graph G with maximum vertex degree Δ. A proper q-colouring of G is an assignment of colours to the vertices of G such that no edge is monochromatic. Calculating the exact number of proper q-colourings of a graph is #P-complete but this number can approximated by sampling from the uniform distribution of proper q-colourings, π, provided that q is sufficiently large. Sampling from π is done by simulating a Markov chain with state space Ω and stationary distribution π where Ω is the set of all q-colourings of G. The mixing time of a Markov chain is how long it takes to get close to its stationary distribution.

We are interested in discovering Markov chains that (1) are mixing in as few steps as possible and (2) succeed in mixing for as few colours as possible. It is known that whenever $q > (11/6)\Delta$, a random update Markov chain (in which, during each step, one vertex is chosen uniformly at random and updated) mixes in $O(n \log n)$ steps (Vigoda, 2000). We study the mixing time of "scan" Markov chains. A Markov Chain is a scan if the vertices of G are visited in an order specified by some permutation which must remain constant during each sweep. It has been shown that scan mixes in $O(\log n)$ steps when $q > 2\Delta$ and in a polynomial number of steps when $q = 2\Delta$ (Dyer, Goldberg and Jerrum, 2005).

In this talk we present a scan which mixes in $O(nlogn)$ steps provided $q \geq 2\Delta$.

Doron Peled, University of Warwick

Efficient model checking for* LTL *with partial order snapshots

Certain behavioural properties of distributed systems are difficult to express in interleaving semantics, whereas they are naturally expressed in terms of partial orders of events or, equivalently, in terms of Mazurkiewicz traces. Examples of such properties are serializability of a database or snapshots.

Recently, a modest extension of LTL by an operator that expresses snapshots has been proposed. It combines the ease of linear (interleaving) specification with this useful partial order concept. The new construct allows one to assert that a global snapshot (also called a slice or a cut) was passed, perhaps not in the observed (interleaved) execution sequence, but possibly in a (trace) equivalent one. A model checking algorithm was suggested for a subset of this logic, with PSPACE complexity in the size of the system and the checked formula. For the whole logic, a solution that is in EXPSPACE in the size of the system (PSPACE in the number of its global states) was given.

In this talk, we present a model checking algorithm which is PSPACE in the size of a system of communicating sequential processes when restricting snapshots to boolean combinations of local properties of each process. Concerning the size of the formula, it is PSPACE for the case of snapshot properties expressed in DNF, and EXPSPACE where a translation to DNF is necessary.

This is joint work with Peter Niebert (Marseille).

Alexis Petrounias, Imperial College London
The small chorded object-oriented language

The chord construct is a concurrency mechanism inspired by the join from the Join-Calculus. Chords were implemented in an extension of C# called Polyphonic C#, and also are available in COmega. They promise to raise the level of abstraction concurrent programs are written in, hence offering a more powerful means of reasoning about the behaviours of such programs.

Furthermore, the inclusion of chords in COmega means they will be used in conjunction with traditional, imperative concurrency constructs such as monitors and threads. In order to study the interactions between other language constructs and chords it is necessary that we fully understand the behaviour of chords. We therefore provided a formal model describing the fundamental semantics of chorded languages, namely the Small Chorded Object-Oriented Language (SCHOOL), to our knowledge the first formalisation of a chorded language.

In this talk I will give an introduction to chords, show what programming with chords looks like, and briefly describe the formal model of chorded programming languages known as Simplified SCHOOL.

Pattarawit Polpinit, University of Warwick
Comparing parallel and sequential selfish routing in the atomic players setting

We consider the problem of routing traffic flow to optimize the performance of a congested network. In particular, we specialize the traffic model with atomic players where each player controls an amount of splittable flow, and aims to minimize their own cost. We are given a network and a linear cost function for each edge. The objective is to compare the optimal social cost with the cost arising from selfish routing decisions by the players.

In this work, we compare a game with two different settings: a parallel setting and a sequential setting. The sequential setting represents a set of flows where all players make decisions sequentially while the parallel setting is a set of flows at Nash equilibrium, i.e. the optimal point for the player given the other players' flows. We prove that in a two-players two-link network model, the social cost of the sequential setting is at most 9/8 times the minimum possible for the parallel setting. We also consider the more general setting in which there are m players in the model.

Sam Sanjabi, University of Oxford
Full abstraction for additive aspects

Aspect-Oriented Programming (AOP) is an emerging programming paradigm that has, from the practitioner's point of view, been extensively studied in the last few years. It allows programmers to "weave" fragments of new code into existing code without modifying the base code. These fragments can potentially exchange data with the base code, or even elide its execution altogether. It has therefore become apparent that current aspect oriented languages such as AspectJ (an aspect-oriented extension of Java) – while providing a powerful programming tool – also break many desirable modularity principles, making it almost impossible to reason about code in the presence of aspects. It seems therefore necessary to subject aspect-orientation to some degree of formal analysis in order to learn to control program behaviour, while retaining as many of the benefits of the paradigm as possible. However, theoretical study – while increasingly active – has lagged far behind implementation in this field.

In this talk, I shall present (to my knowledge) the first known fully abstract game semantics for a non-trivial fragment of AOP: additive aspects. Additive aspects are those which do not prevent the base computation from executing, but may otherwise exchange information with it in any way. The full abstraction result is achieved (in the style of McCusker) by demonstrating the existence of a compositional, fully abstract translation between a simple language of additive aspects, and Idealised Algol with General References (an imperative language known to have an existing games model). After presenting the main theorem, I discuss some the implications, applications, and potential extensions of the result.

Paul Sant, University of Luton
Combinatorics of colouring 3-regular trees

An instance of the Colouring Pairs of Binary Trees Problem (CPBT) consists of two 3-regular trees, both with n leaves. The challenge of CPBT is to show that, for every instance of the problem, there exists 3-edge-colourings of the trees such that the sequence of colours associated with root-edges in both trees is the same. Interest in the problem stems from its equivalence to a number of other important combinatorial problems including, specifically, the historically-famous 4-colour problem of planar maps. We present a number of recent results, both combinatorial and algorithmic, related to CPBT and pose a number of challenges.

This is joint work with Alan Gibbons (King's College London).

Rahul Savani, London School of Economics
Hard-to-solve bimatrix games

A bimatrix game is a two-player game in strategic form, a basic model in game theory. A Nash equilibrium is a pair of (possibly randomized) strategies, one for each player, so that no player can do better by unilaterally changing his or her strategy. The problem of finding one Nash equilibrium of a bimatrix game is considered as "one of the most important concrete open questions on the boundary of P today" (Papadimitriou, 2001).

In this talk, we show that the commonly used Lemke-Howson algorithm for finding one equilibrium of a bimatrix game is *not* polynomial. This question had been open for some time. The algorithm is a pivoting method similar to the simplex algorithm for linear programming. We present a class of square bimatrix games for which the shortest Lemke-Howson path grows exponentially in the dimension d of the game. We construct the games using pairs of dual cyclic polytopes with $2d$ facets in d-space.

This is joint work with Bernhard von Stengel.

Anton Setzer, Swansea University

Inductive recursive definitions and generic programming

Inductive-recursive definitions were originally introduced by Peter Dybjer in order to capture the general principle for defining sets in Martin-Löf type theory. They generalise the notion of a strictly positively inductively defined set by allowing to define a set inductively while simultaneously defining recursively an element of another type (which could be a set, the type of sets, or a higher type like the type of functions mapping sets to sets). Together with Peter Dybjer, the author has developed a closed formalisation of inductive-recursive definition, which gives rise to a condensed definition of inductive-recursive definitions. Although inductive-recursive definitions were originally introduced in the context of dependent type theory, their relevance seems not to be restricted to dependent types.

In the specific case where the recursively defined object is a set we obtain a relative general notion of generic functions. The set of elements of the inductive-recursively defined set are codes for data types, and the recursively defined set determines for each code the data type it denotes. A generic function can take a code for a data type and an element of the data type it denotes, and compute from it a code for another data type and an element of the data type it denotes, where the target code could be given by an inductive-recursive definition different from the the one used by the arguments. Some examples of generic functions will make use of inductive-recursive definitions of higher types.

In this talk we introduce the notion of inductive-recursive definitions and give some examples of generic functions definable using inductive-recursive definitions.

Nikolaos Siafakas, King's College London

A fully labeled lambda calculus

Levy's labeled lambda calculus has been of major importance in the research of efficient implementations of functional programming languages. If we compute the normal form of a term then the resulting label will describe a path in the graph of the term. The Geometry Of Interaction (GOI) machine, which is an implementation of Girard's Geometry of Interaction semantics for Linear Logic, will follow exactly the path described by the label. The investigation of the structure of the labels has led to optimised versions of the GOI machine, such as the Jumping Abstract Machine (JAM), where the length of the path to be traversed is significantly reduced. However, the structure of the labels is different from the structure of the paths. The latter depend on the choice of translation of the lambda calculus into Linear Logic proof nets (call-by-value or call-by-name) and the optimisations rely on the transposition of the structural information obtained from the labels into paths. For each translation we present a fully labeled lambda calculus which unifies the information yielded from the paths with the structure provided by Levy's labels. Our main goal is to find new and efficient ways of computing the paths in the GOI machine.

Alexandros Skaliotis, King's College London

Logarithmic simulated annealing for protein folding

Protein folding is the process by which a sequence of amino-acids conforms to a three-dimensional shape that determines the biological function of the resulting protein. We consider the problem of predicting these conformations based on the hydrophilic-hydrophobic model introduced by Dill et al. in 1995. A problem instance consists of a chain of amino-acids, each labeled "H" (hydrophobic) or "P" (hydrophilic). This sequence has to be placed in a 2D or 3D grid in a non-overlapping way that preserves the original adjacencies of the amino acids. Following Anfinsen's hypothesis which suggests that proteins fold to a minimum energy state, the goal is to minimise the overall energy. In the simplest variation, this corresponds to maximising the number of adjacent hydrophobic pairs. The protein folding problem in the HP model is NP-hard in both 2D and 3D. In 2004, Fu and Wang gave an $\exp(O(n^{1-1/d})ln(n))$ algorithm for d-dimensional protein folding simulation in the HP-model.

We investigate the application of logarithmic simulated annealing to the problem by employing a set of moves proposed by Lesh et al. in 2003 and Blazewicz et al. in 2005. Albrecht et al. in 2006 show that after $(n/a)^{O(G)}$ Markov chain transitions, the probability of being in a minimum energy conformation is at least $1-a$, where n is the length of the instance and G is the maximum value of the minimum escape height from local minima of the underlying energy landscape. For selected benchmark instances we performed an experimental estimation of values for G. These indicate that $G < n^{1-1/d}$ which is competitive to the bound by Fu and Wang.

This is joint work with Andreas Albrecht (University of Hertfordshire) and Kathleen Steinhofel (King's College London).

Colin Sng, University of Glasgow

Popularity in the capacitated house allocation problem

We consider the problem of finding a popular matching in the Capacitated House Allocation problem (CHA), in which we have a set of agents and a set of houses. Each agent has a preference list ranking a subset of houses in some order of preference, and each house may be matched to a number of agents that must not exceed its capacity. A matching M is popular if there is no other matching M' such that the number of agents who prefer their houses in M' to M exceeds the number of agents who prefer their houses in M to M'. Here, we give a polynomial-time algorithm to determine if an instance of CHA admits a popular matching, and to find a largest such matching if one exists. Specifically, the complexity of the algorithm is $O(r^{3/2}s^{1/2})$ where r is the number of agents and s is the number of houses.

Mike Stannett, University of Sheffield

Future trends in hypercomputation

Hypercomputation (the theory of "super-Turing machines") is a rapidly expanding area of Theoretical Computer Science, with links to physics, philosophy, biology and mathematics. We present a user-friendly guide to the current state-of-the-art, including quantum computation and cosmological models, and suggest a number of theoretical problems whose solution might prove important to the future development of the subject.

Wouter Swierstra, University of Nottingham
Isomorphisms for context-free types

We can show two types to be isomorphic by constructing explicit coercion functions. What should we do if we cannot find such an isomorphism? There's no way to be sure we're not looking hard enough. I briefly show how a variation of classical monadic parser combinators can be used to provide evidence that two inductive types are not isomorphic.

Traditionally, monadic parser combinators consume an input sequence of symbols. Instead of input strings, however, we consider parsing multisets. Interestingly, we can then interpret the results of our parsers as a powerseries. We can show that two types are isomorphic if and only if their associated powerseries are identical. Distinguishing non-isomorphic types now simply reduces to finding a disparity between two powerseries.

Rick Thomas, University of Leicester
FA-presentable structures

A structure consists of a set together with a collection of relations. For example, a group consists of a set together with a ternary relation (representing the composition of elements in the group), a unary relation (yielding the identity element) and a binary relation (representing the process of taking the inverse of an element). A natural general question is the following: given a structure, can we perform computations in it?

The natural approach would be to take some general model of computation such as a Turing machine. A structure would then be said to be computable if its domain can be represented by a set which is accepted by a Turing machine and if there are decision-making Turing machines for each of its relations. However, there have been various ideas put forward to restrict the model of computation used; whilst the range of structures decreases, the computation can become more efficient and certain properties of the structure may become decidable.

One interesting approach was introduced by Khoussainov and Nerode who considered structures whose domain and relations can be checked by finite automata as opposed to Turing machines; such a structure is said to be "FA-presentable". This was inspired, in part, by the theory of "automatic groups" introduced by Epstein et al; however, the definitions are somewhat different.

We will report on some recent results obtained in conjunction with Graham Oliver and Andre Nies. In particular, we will survey some of what is known about the possible structure of FA-presentable groups and rings.

Ashutosh Trivedi, University of Warwick
Average time games

An average time game is played on the infinite graph of configurations of a finite timed automaton. The two players, Min and Max, construct an infinite run of the automaton by taking turns to perform a timed transition. Player Min wants to minimize the average time per transition and player Max wants to maximize it.

In this work the strategy improvement algorithm for average payoff games on finite graphs is generalized to solve average time games. A direct consequence is an elementary proof of determinacy for average time games. This generalizes results of Asarin and Maler for optimal time reachability games and it partially solves a problem posed by Bouyer et al., to design an algorithm for solving average payoff games on priced timed automata.

Zheng Wang, University of Manchester
A component model for verified software

Component-Based Software Development (CBSD) is a promising approach for assembling pre-existing software components into an integrated software system; this potentially helps to achieve effective software reuse, easy software evolution a and low cost-to-performance ratio. Software verification is a long-standing grand challenge. The aim is to verify software correctness using theorem provers or model checkers. This task is practically impossible for large software systems, for either technical or cost reasons, and is currently not done except for small to medium-sized safety-critical applications. Our research focuses on cross-fertilisation between these two areas: how to make the verification task practically feasible by decomposing and devolving it to software components.

Cross-fertilising CBSD with software verification could offer a practical way to construct verified software from pre-verified components. The central issue is how to design a CBSD methodology that can reuse component proofs via composition operators and the cornerstone of such CBSD methodology is its underlying software component model that provides the semantic framework of components. In order to tackle this issue, we build a software component model that defines the semantics and syntax of software components and their composition. In our component model, components can be built, verified and stored in a repository. Larger components can then be composed from these components using composition operators that carry proof plans that reuse the existing proofs of the components. Thus verification of large systems can be decomposed and devolved to the components, i.e. it can be done by composing or reusing component proofs via proof plans associated with the composition operators.

In this talk I give an overview of our component model and show how it helps to construct verified software from pre-verified components, and demonstrate the feasibility of our component model with an industrial case study: an automatic train protection system.

Taoyang Wu, Queen Mary, University of London
Fixed point free property is* NP*-complete

Considering a permutation group G naturally acting on a finite set X, an element $g \in G$ is called ***fixed point free*** if it doesn't fix any point in X. A natural problem is whether there is any fixed point free element for given G and X. When G is input as a set of generators, we show that the problem is NP-complete via reduction from 3-SAT problem. To this end we also present another NP-complete problem: the solvability of a certain type of incongruences.

This is joint work with Peter Cameron.

Yonghong Xiang, Durham University

Fault-tolerant properties of k-ary n-cube

First, some introduction of fault-tolerant properties of interconnection networks will be introduced. Then, our consideration of finding a longest fault free path in a faulty k-ary n-cube with at most $2n-2$ faults for even integer k and $n \geq 3$ will be given.

Hong Qing Yu, University of Leicester

Semantic web services composition via planning as model checking

The ability to automatically compose services is one essential aspect of service oriented architecture. It can reduce time and cost in development and maintenance of complex services and software systems. We are developing a technique to realize this aim by combining the "planning as model checking" approach with semantic web service concepts. We have modified a planning as model checking algorithm by using a bounded on-the-fly depth-first search algorithm that its possible service execution plans are generated on the fly. One of the challenges is to model a web service as a state transition system. The approach will be suitable in the context of ontologies, but for now we are simply using dictionaries for mapping operations and parameters. The planning as model checking approach forms part of a larger framework to automatically compose services, which addresses several drawbacks of current composition approaches.

Michele Zito, University of Liverpool

Lower bounds for dominating sets in web graphs

In this work we study the size of generalised dominating sets in graph processes that model aspects of the World Wide Web. We show that graphs generated in this way have fairly large dominating sets (i.e., linear in the size of the graph). Various results will be described that enable us to prove increasingly good bounds. The proof techniques we present may be applicable to the study of other combinatorial properties of web graphs.

This is joint work with Colin Cooper (King's College London) and Ralf Klasing (Bordeaux).