REPORTS ON CONFERENCES

Report on BCTCS 12 at the University of Kent, 1996

The 12th Annual Meeting of the British Colloquium for Theoretical Computer Science, took place at the University of Kent from 1-4 April 1996. The colloquium was held in Darwin College, one of the four University colleges, and located close to the historical cathedral city of Canterbury in the south east of England.

The meeting followed the, now established, form of a small series of invited one hour presentations and a number of shorter research presentations. The meeting was attended by around 90 participants, with a total of 45 contributed talks being given over the 3 days. There were invited talks presented by: P. Aczel (Manchester), G. Birtwistle (Leeds), E. Brinksma (Twente), F.K. Hanna (Kent), U. Martin (St. Andrews), M. Smid (King's College, London), J. Toran (Ulm), and D.A. Turner (Kent).

The colloquium was well-organised by Simon Thompson and John Derrick and further established BCTCS as one of the principal annual events for U.K. based theoreticians. Generous financial support for 25 Ph.D students was provided the EPSRC.

BCTCS 13 will be held at the University of Sheffield between 23-26 March 1997. Information concerning this may be found at http://www.dcs.shef.ac.uk/~wmlh/BCTCS13.html.

Unrefereed contributions describing ongoing research from both U.K. and overseas research students and academics are warmly encouraged for BCTCS 13, and BCTCS14 which will be hosted by the University of St. Andrews. Further background on the BCTCS may also be found at http://www.csc.liv.ac.uk/~ped/bctcs/summary.html.

Paul E. Dunne (Liverpool)
Simon J. Thompson (Kent)

BCTCS 12
British Colloquium for Theoretical Computer Science
1-4 April 1996

Abstracts of invited talks

Formalising Abstract Algebra in Constructive Type Theory
Peter Aczel

Today there is considerable worldwide activity in the design, development and use of a great variety of computer systems in which the correctness of mathematical assertions can be machine checked, generally with the interactive help of the user. A major motivation for this activity has been the desire for the efficient production of correct software, together with the feeling that computers might help in the correctness checking. It is controversial whether it can be generally worthwhile and feasible to get the correctness of software to be machine checked, even with interactive human help. Clearly it will be necessary to build up libraries of machine checked mathematical theories, which can be conveniently accessed by the software developer.

In my talk I will review my own approach to the intellectual challenge to design systems and methods that can make the large scale production of machine checked formalised mathematics feasible. I will focus on the formalisation of abstract algebra in constructive type theory using the LEGO computer system.

Specifying and Verifying AMULET in CCS
Graham Birtwistle

Asynchronous hardware design is attracting renewed research and industrial interest because of its compositional properties and potential for low power consumption (much of today's hardware is battery driven and asynchronous circuits only do work when there is work to do).

AMULET is an (already working) asynchronous version of the ARM micro designed by Steve Furber's team at Manchester University. ARM is now a standard TI cell, used by IBM to control disks, and in the Apple Newton. Its outstanding characteristic is perhaps its very low power consumption (132 MIPS per watt). The medium term expectation is that AMULET will fare even better.

The talk will present work completed on specifying and verifying the major subsystems of AMULET (address interface, register bank, execution pipeline, data interface) in CCS and modal mu respectively at the RTL. CCS does not handle data well, so we concentrated our efforts on the way blocks (a)synchronise. Once the major subsystems were specified they were tested for deadlock and livelock, bus contention, etc. using modal mu.

The work was carried out with Ms Ying Liu (now at BNR) and with the full cooperation of the Manchester AMULET team.

Formal Models for Testing:
An Interaction between Theory and Practice
Ed Brinksma

For a number of years now, we have investigated the possibility of applying the process-algebraic theory of testing pre-orders to the practical problems of actual testing, most notably to the problem of test generation. This has turned out to be a very interesting area of research, where the interaction between theory and practice has led not only to better algorithms and tools to derive test suites from specifications, but also motivated new theoretical developments. In our presentation we will give an overview of the research in this area, presenting its historical development, the main results and the open problems.

Reasoning about Digital Systems
Keith Hanna

Contemporary digital systems, with hundreds of thousands of gates to a chip (and large recall costs!), are a prime area for the application of formal methods.

I will give an overview of the use of higher-order logic for describing and reasoning about digital systems and then focus on the gains, and drawbacks, offered by the use of dependent types.

The princess and the plumber, the role of
mathematics in computer science.

Ursula Martin

Spanners: approximating the complete Euclidean graph
Michiel Smid

Let S be a set of n points in the plane, and let $t > 1$ be a real number. A *t−spanner* is a sparse graph having the points of S as its vertices, such that for any two points p and q in S, there is a path between them of length at most t times the Euclidean distance between p and q.

Spanners are important data structures in geometric algorithm design, because they provide a means of approximating the complete Euclidean graph with only $O(n)$ edges. In many applications of spanners, it is important that the spanner possess a number of additional properties, such as low total edge weight, bounded degree, and low diameter. Such spanners have much better properties than minimum spanning trees, which have been used extensively to approximate the complete Euclidean graph.

In this talk, I will give an overview of recent results on constructing spanners.

On the complexity of the Graph Isomorphism problem
Jacobo Toran

The Graph Isomorphism problem is one of the few problems in the class *NP* that is not known to be complete nor polynomial time solvable. Because of its theoretical and practical importance the problem has been studied from many different approaches.

This tutorial will present several results that provide a better understanding of the relative position of the Graph Isomorphism problem in *NP* and other complexity classes. The GI problem will be used to illustrate important concepts in Complexity Theory like reducibilities, interactive proofs, counting properties, enumeration of the solutions or circuit complexity.

Total Functional Programming
David Turner

The driving idea of functional programming is to make programming more closely related to mathematics. A program in a modern functional language, such as Haskell or Miranda, consists of equations, which are simultaneously computation rules and axioms for reasoning about the entities thus computed.

In this talk I will argue that our existing model of functional programming, although elegant and powerful, is mathematically compromised - to a much greater extent than is commonly recognised - by the presence of partial functions. I will sketch a proposed discipline of total functional programming, based on introducing a sharp distinction, enforced by the type system, between data and codata.

Abstracts of contributed talks

Machine-Assisted Meta-Theoretic Proof
Andrew A Adams

Formal systems of logical entailment are one of the fundamental tools of theoretical computer science. Many logical frameworks exist to implement proof assistants and automated theorem provers for such systems, and many such implementations exist, yet little work has been done on machine assistance for proof about formal systems. Preliminary results of a project comparing four logical framework systems (Alf, Sequel, Isabelle and Coq) in their suitability for this work are presented. Ongoing work with Coq, the system found to be most suitable is also presented, together with some comments on the general problems in work of this nature.

Implementing Rewriting as a Base for Building
Intelligent Tutorial System
Dhiya Al-Jumeily

The word problem in universal algebra is the problem of deciding whether two words composed of variables and operators can be proved equal as a consequences of a given set of identities by the operators. We will construct an algorithm, called the "Valley algorithm", which uses simplification ordering, for the aim of guiding an intelligent tutoring system for algebra. The basic problem is to examine each step of the student solution and compare it with lecturer solution. Each step will be represented as a term, and these terms are subject to rewrite rules. Graphical tool has been provided to construct the rewrite terms.

A New Model of DNA Computation
Martyn Amos

This paper introduces a new model of computation that employs the tools of molecular biology whose practical implementation is far more error-resistant than extant proposals. We describe an abstraction of the model which lends itself to natural algorithmic description, particularly for problems in the complexity class NP.In addition we describe a number of linear-time algorithms within our model, particularly for NP-complete problems. We describe an in vitro realisation of the model and conclude with a discussion of future work.

Mathematical Programming with Functional Languages
Chris Angus

Functional programming languages such as Haskell allow numerical algorithms to be expressed in a concise, machine-independent manner that closely reflect the underlying mathematical notation in which the algorithm is expressed. Unfortunately the price paid for this level of abstraction is usually an increase in execution time and space usage.

Ongoing work towards the goal of developing efficient large-scale numerical software in a functional language will be described in two stages:

Use of the purely functional language Haskell for the development of a (problem/dimension)-independent finite element analysis system and the advantages of working in this style, focusing on such areas as the usefulness of Haskell-style type classes and the appropriateness of non-strict semantics in this area.

Lessons learned from experiences described above which contribute to design decisions,implementation techniques and necessary multi-parameter extensions to Haskell-style type classes for a functional language specialized in the area of numerical programming.

Extending the syntax of a type theory with a system of implicit coercions to ease the development of formal mathematics

I would be likely to be talking about using implicit coercions to bring a formal presentation of mathematics, in particular some algebra, rather closer to the more readable informal style that non-formal mathematicians would be used to. (I have recently extended the proof-checker LEGO with such coercions.)

Iterative construction of data modelling language semantics.
Richard Botting

A methodology for providing a formal description of large data modelling languages has been considered. It is based on the idea that languages can be described by a chain of sub-languages such that each sub-language is capable of describing the semantics of the next sub- language in the chain. Transformation functions can be written at each stage, mapping models written in the sub-language to a target language such as VDM-SL. The source of this mapping will be a VDM-SL version of the sub-language model. The suggestion is that the consistency of this system can be tested by inputting the sub-language meta-model itself. This should provide a second VDM-SL version isomorphic to the first. Investigation of whether these two models are independent of one another needs to be carried out. The paper discusses a case study application.

A Denotational Semantics for Real-Time LOTOS
Jeremy Bryans

LOTOS is a Formal Description Technique developed during the years 1981-1986. It became an ISO international Standard in 1988. It consists of a behavioural part, based on the process algebras CCS and CSP, and a data part, based on equational specification of data types.

The behavioural part has no direct way of describing timing properties, and much work has gone into addressing this issue. The various proposals are now converging.

We present one of the proposed timed extensions, and use it as a vehicle to investigate the problems in giving a denotational semantics to a timed language. We propose enhancements to a standard (trace,refusal) semantics that overcome these problems. The advantages of a denotational semantics are then discussed.

Approximating the volume of convex bodies: a new approach.
Russ Bubley

We present a fully polynomial randomized approximation scheme for the volume of a convex body in Euclidean n-space. Previous algorithms for computing volumes have relied on complicated conductance arguments and isoperimetric inequalities; we show that the conceptually simpler coupling argument can be used to give a similar result. (Joint work with Martin Dyer and Mark Jerrum.)

Modelling Discrete Distributions in WSCCS

Supporting Reasoning about Functional Programs : An Operational Approach

Athina Christodoulou

It is often stated that one advantage of functional programming languages, in particular lazy functional languages, is that they are suitable for formal reasoning. While such reasoning is indeed carried out by some, there is a need for tools to make it more practical and usable by a wider community.

We discuss how a reasoning system can be built, within the HOL theorem proving environment, based on an operational semantics for the language. Applicative bisimulation is used to define program equivalence. This allows reasoning using both induction and co-induction. We discuss how this theory can be embedded in HOL and the kind of tools which can be built on top of this theoretical framework to make reasoning practical.

Average Case Complexity Measures for Boolean Functions

Paul E. Dunne

The classical theory of the complexity of Boolean functions has tended to concentrate on worst-case concepts of function complexity within the divers models of computation appropriate to this domain. This is unsurprising since the measures typically examined with these models, circuit size and circuit depth, do not appear to have very 'natural' formulations in contexts other than worst-case complexity. Recently, definitions for the 'average' case complexity of Boolean functions with respect to circuit depth have been proposed. In this talk, we extend the concept of average case complexity to circuit and formula size, and discuss how such measures may be related to complexity measures for Boolean-functions arising from models of demand-driven circuit simulation.

The complexity of algorithms for content-addressable memory

Jason Emmett

Content-addessable memory (CAM) is a memory storage device that accesses its data by content rather than by address. CAMs can also be thought of as a simple parallel computer where each processing node consists of a memory word and a comparator. Although their computational power is low the large amount of processing nodes that can be placed on a single chip still makes them of interest in the real world. CAMs have previously been limited by their capacity but advances in hardware technology have meant CAMs of a more usable size are now available.

Little research has taken place into the complexities od the algorithms that run on CAMs. In this talk we give some background to the different types of CAM and present some algorithms. One of these will be sorting and a detailed discussion of the complexity of the algorithm under varying input conditions will be discussed.

An intensional completeness theorem that connects ternary simulation with timing analysis

Matt Fairtlough

We prove the equivalence between the ternary circuit model and a notion of intuitionistic stabilization bounds. The results are obtained as an application of Mendler's timing interpretation of intuitionistic propositional logic. We show that if one takes an intensional view of the ternary model then the delays that have been abstracted away can be completely recovered. Our intensional soundness and completeness theorems imply that the extracted delays are both correct and exact; thus we have developed a framework which unifies ternary simulation and functional timing analysis. Our focus is on the combinational behaviour of gate-level circuits with feedback.

Towards a Hardware Description Framework for Asynchronous Micropipeline Design

Donal Fellows

There has recently been a renewed interest in asynchronous digital circuit design. The analysis of the behaviour of these circuits is, however, rather more complex than the usual synchronous systems.

For formal methods to help the design engineer, formalisms must get embedded into systems that can support the way the engineer thinks about her design, which may be presented as a mixture of descriptions at many different levels of abstraction. One particular goal with our approach is the development of application specific verification methods that can potentially gain extra purchase on the high complexity of formal verification. This, in turn, can require formalisms to closely match the levels at which designs are thought about.

In this talk, I will discuss various elements of the 'Rainbow' HDL, designed to address this; I will concentrate on the fragment for describing designs in a dataflow style.

On Minimising Automata and Congruence Closure in Parallel
Clive Galley

Minimisation of automata, and congruence closure have been shown to be equivalent to the k functions coarsest set partition problem, KFSP. For a set S such that $|S|=n$ and k functions from S to S, the equivalent automaton has n states, and a k character alphabet.

We present some new bounds for KFSP on the CRCW PRAM model of computation.

A Verification of Parallel Associative Combinator Evaluation using Pi Calculus.
Tim Glover

PACE is a design for a parallel combinator graph reduction machine. Starting from a pi calculus description of combinator reduction, a series of valid refinements can be made culminating in a model of the PACE architecture. This motivates some minor but useful modifications to the calculus, and suggests that it may be useful to make use of different notions of satisfaction at different points in the refinement.

Learning Foraging Thresholds for Lizards
Leslie Goldberg

This work gives a proof of convergence for a randomized learning algorithm that describes how anoles (lizards found in the Carribean) learn a foraging threshold distance. The model assumes that an anole will pursue a prey if and only if it is within this threshold of the anole's perch. This learning algorithm was proposed by the biologist Roughgarden and his colleagues. They experimentally confirmed that this algorithm quickly converges to the foraging threshold that is predicted by optimal foraging theory. Our analysis provides an analytic confirmation that the learning algorithm converges to this optimal foraging threshold with high probability. (joint work with William E. Hart of Sandia National Labs and David Bruce Wilson of MIT)

Synthesising synchronisation in the Petri Box Calculus.
Martin Hesketh

The box algebra can be considered a variant of CCS, with the synchronisation operation extended to provide multi-way synchronisation. The semantics of the box algebra are given in terms of Petri boxes (labelled Petri nets). A Petri box can be constructed compositionally from a given box expression. The process of finding a box expression for a given Petri box is known as synthesis. This talk describes a solution to the synthesis problem for a fragment of the box calculus containing synchronisation. In particular, an algorithm is presented which deals with transitions in the Petri box which have arisen from a synchronisation operator. A by-product of the algorithm is the definition of a normal form for box expressions involving synchronisation.

Partial evaluation of parametrised circuits for dynamic hardware
Jonathan Hogg

A common technique in functional programming is to partially apply functions to one or more parameters giving a reduced function which can be applied to the remaining data. In implementation terms this results in smaller faster programs. The same technique can be used with hardware description languages to reduce the size and delay of a circuit given some constant parameters. Normally this would be of limited use as few if any parameters can be determined in advance for a static circuit. However, new dynamic hardware (such as FPGAs) makes possible the implementation of circuits at run-time, thus allowing the generation of a circuit with run-time parameters. We apply this concept to the relational hardware description language Ruby. The formal nature of Ruby provides a basis for reasoning about such hardware transformations, in contrast to the more ad-hoc manner in which common hardware synthesis systems optimise circuits.

Hybrid Systems
Mike Holcombe

Safety-critical systems are used as a motivation for much research in formal methods. It is claimed that to ensure correctness and safety such systems should be formally specified and verified. We examine to what extent existing formal methods, notations and languages are suitable for this task. In particular we look at the essential common features and issues involved in the design of safety-critical systems and propose a new model and semantics for their representation and analysis which tries to address many of these issues.

An Operational View of Visual Languages
Chris Holt

The graph grammar approach to defining visual languages is not very intuitive; it reflects neither the way programs are constructed, nor the way they are read and understood. For instance, when three mappings A, B and C are composed such that one result of A goes to B, one result of B goes to C, and a second result of A goes to C, the compositions are non-hierarchical and do not naturally reflect a series of productions.

When constructing textual programs, the editing grammar is less restrictive than the language grammar, but is more easily understood; the production rules are simply editing operations. It is suggested that such an approach might be helpful for visual languages: the actual editing instructions comprise the rules that define the language. These consist of the creation, modification and destruction of objects and links; they also describe allowed behaviours of objects in manipulating other objects.

Existence and Intuitionistic All-Elimination
Alan Hutchinson

Traditional higher order intuitionistic logic can be expressed in a system with sorts, variables, implication "=>" and the existence symbol "E" of Fourman and Scott. This system is succinct. It makes some subtleties in the Brouwer-Heyting-Kreisel explanation of intuitionism appear unnecessary, and the Curry-Howard correspondence appear natural. The All-Elimination rule needs a slight revision, which also appears natural. The system has a simple notion of model in toposes. There seem to be different notions of constructibility among the standard intuitionistic inference rules.

An Action Calculus for Functional Programming
Ole H. Jensen

Milner's action calculi are intended primarily as a common framework for models of interactive computation. Here we show how to use an action calculus to express the operational semantics of a functional language, and how to extend the semantics to include references. This serves as an illustration of the general concept of action calculi, and it also suggests how action calculi are able to describe the various features of a language in a modular way.

Adequate encodings of proof systems in UTT.
Nikos Mylonakis

In this talk, first I will give an overview of metalanguages and the type theory UTT (Uniform Theory of dependent Types), then I will talk about sound and complete proof systems for behavioural specifications and finally I will provide a sketch of the encoding (and its proof of adequacy) of such a proof systems in UTT.

Two concepts of possible worlds in probabilistic reasoning
Rashmi Pandya

Higher-order probability has been proposed as a method for reasoning about uncertainty in AI. Various existing approaches are based on one of two possible worlds models. This talks describes propositional worlds models and probability worlds models in the context of these probability systems and examines the distinctness and applicability of the two models. I will show how the analysis of higher-order systems at the possible worlds level can provide a more concise description of these systems and provide the steps needed towards a general theory of higher-order probability for reasoning in AI.

Structure from String-Folding
Mike Paterson

Prediction of the three-dimensional structure of a protein from its amino acid sequence is an important, but difficult, problem. In our forthcoming ICALP paper, Teresa Prztycka and I consider a simple discrete analogue of the protein folding problem. A string of symbols over some alphabet is to be embedded in the three- or two-dimensional grid so as to maximise the number of adjacencies between equal symbols. We show there that finding an embedding with a maximal number of such matches is NP-hard.

The most challenging part of the proof lies in designing strings for which we can prove that their optimal embeddings form particular structures. In this talk I shall show some of these constructions and set out some challenging open problems.

Timed Interaction Categories
Justin Pearson

This talk will present on-going work on timed interaction categories. Timed Interaction categories extend extend Abramsky's program of types for concurrent systems, to timed process algebras. I will present two categories one for synchronous and probabilisitic processes and a category which models Wang's timed extension of CCS.

On Transformations of Concurrent Object Programs
Anna Philippou

The pi-calculus is a process calculus in which concurrent systems with evolving structure can be naturally expressed. Its basic theory is now well established and it has been used to model and reason about a variety of systems. In particular, the pi-calculus and several extensions of it have been used successfully to give natural and tractable semantic definitions for concurrent object programming languages.

In this talk, we consider a concurrent object language and its translational semantics to a mobile process calculus. We then present transformation rules which increase the scope of concurrent activity within systems prescribed by programs without altering their observable behaviour. A central concern is to prevent sharing of references to objects of some of a program's classes by imposing syntactic conditions. The correctness of the rules is proved by using the semantic definition and the notions of confluence and partial confluence play a significant role.

Matricial Space Economy with Constant Access Time
Ida Pu

In many applications involving sparse matrices there is a potential for economic use of storage space by suitable matricial compression. This could simply involve storing only the significant entries of the matrix along with their original addresses. However, such a strategy does not allow access to an original matrix element (from its address subscripts) by a sequential algorithm in constant time, although a logarithmic time algorithm is very easy to devise. We describe another, very simple, strategy for matricial compression which allows access to any original matrix element in constant time and we analyze the average saving in storage space.

Event structures, domains and automata
Vincent Schmitt

Correspondences between event structures, domains and automata with concurrency relations have been widely studied (by Winskel, Bednarczyk, Droste, Panangaden, Stark and Kuske). We concentrate on "recognizable" domains which are unfoldings of finite automata and compare the various notions of recognizabilty induced by distinct types of automata. For this purpose we introduce regular labellings of event structures. We show that a reflection of the category of event domains into a subcategory with dI-domains as objects does not preserve recognizability.

We prove that the class of domains unfoldings of finite stable trace automaton and the one of unfoldings of finite asynchronous systems coincide. Many problems remain unsolved : Do the conflict event domains unfoldings of finite concurrent automaton correspond with unfoldings of finite trace automaton? What are the domains unfoldings of finite nets? Is there any interesting order theoretic characterization for recognizable domains?

An Application of Type Theory to Semantic Networks.
Simon Shiu

Semantic Networks are a graph based knowledge representation with associated reasoning mechanisms. Intuitively they have many desirable properties, such as efficiency, naturalness and expressiveness. However showing this formally has been problematic.

Constructive type theory with the 'proposition as types' principle, due to Curry and Howard, has an internal logic. Proof assistants based on such theories have been built to support work in constructive mathematics and formal methods. This framework can be used to define and reason about 'abstract theories' in a modular way.

This talk will describe how such a theory has been defined for SemNet (a semantic network used at Durham). Some of the features of type theory (intensionality and dependent types) will be shown to be well suited to describing semantic networks.

Formal Software and Hardware Design using Algebraic Methods
L. J. Steggles

Algebraic specification methods are a theoretically well-founded formal method for computer system design, supporting formal specification, verification and program refinement. However, in order for algebraic specification methods to be accepted as a practical formal method further research is needed into the methodology of using these techniques in practice and the integration of software tools into the formal design process. In this talk we propose a simple methodology for the formal verification of computing systems using (higher-order) algebraic specification methods. This verification methodology is based on the notion of functional refinement and we define the conditions necessary for a design specification to be a correct functional refinement of an abstract requirement specification. In particular we consider the role of the environment information contained within the requirement specification during the verification process. We demonstrate our verification methodology by considering the formal verification of a systolic algorithm for computing the convolution function using the CSDM software development system.

SOFL: A Formal Engineering Methodology for Industrial Applications
Yong Sun

There is a strong challenge to formal methods that they cannot be easily and widely adopted in industry. A major reason for this is that they are difficult to use and their application consumes prohibitive amounts of resource. Much research on the integration of available formal methods (e.g. Z, VDM, B-method) and either Structured Methodology or Object-Oriented Methodology has been conducted to make formal methods more practical, but its success has been very limited. However, no effort has been made to integrate properly formal methods, structured methodologies and object- oriented methodologies to take advantage of the desirable features of the three approaches.

As one approach to the solution of these problems, we propose a language called SOFL (Structured Object-oriented Formal Language) for system development. It supports the concept that a system can be constructed using the structured methodology in the early stages of its development, and by using object-oriented methodology at later, more detailed levels. During the complete system development process, formal methods are applied in a manner that best uses their capabilities.

Formal Methods for Accident Analysis
Alastair Telford

Reports of major accidents, such as the fire at Kings Cross or the explosion at Piper Alpha, do not always present the complex interactions that led to a disaster in a precise and unambiguous manner. This is mainly due to the fact that particular views upon the accident, such as given by human factors specialists, oil platform engineers etc., are separated into distinct chapters within an accident report.

It is vital for future safety that such reports are precise and their findings unambiguously disseminated. To achieve this we have been investigating the use of formal methods which have previously been applied mainly to software engineering projects. We present our work in this area, particularly with regard to temporal logics, and show how such formal descriptions may be used to detect inconsistencies, omissions or superfluous detail. In addition, we describe the application of cross-viewpoint consistency to accident analysis.

Finite Presentability of Semigroups
Rick Thomas

This talk will describe some recent work done with Colin Campbell, Edmund Robinson and Nikola Ruskuc from the University of St Andrews. Given a finitely-presented semigroup, are there conditions for certain subsemigroups to be finitely-presented, and are there methods for finding these presentations? The talk will survey some recent results, such as the fact the a right ideal of finite index in a finitely presented semigroup is itself finitely presented. If we add the hypothesis that our semigroup is free, then one can prove stronger results, such as the fact then any right ideal which is finitely generated (as a semigroup) is necessarily finitely presented.

Efficiently Composing Distributions for Performance Modelling
Chris Tofts

There has been a great deal of attention to the production of formal performance models using process algebras or petri nets to describe the underlying systems. In this talk we address the issue of composition at the level of the underlying probability distributions of the components in the system. This is of particular importance if we wish to derive system properties for large scale practical problems. We describe several abstraction techniques and their limiting behaviour with respect to performance parameters such as mean time cost.

Animating Specifications via Model Generation
Colm Toomey

Animation is used to identify flaws in requirements specifications without resorting to executable specifications or requiring a decision procedure. Attempts have been made to animate algebraic specifications using prolog. However, prolog suffers from several well-known drawbacks due to its procedural semantics. In addition prolog lacks support for the equality relation, and is restricted to horn clauses.

Classically, logic and functional programming are integrated by providing narrowing as the single inference rule. Instead we combine the model-generation prover of Manthey and Bry with a ground ac-completion algorithm used to obtain canonical rewrite systems for a given collection of (ground) equations. The resulting prover is able to handle a restricted class of FOPC clauses with equality and functional terms, and does not suffer from many of the usual prolog drawbacks.

We report on the application of the extended model generation prover to the problem of animating algebraic specifications of requirements, using the telephone exchange problem of Loomes and Woodcock as a case study.

Using OBDD encodings for Space Efficient State Storage
during On-the-fly Model Checking
Willem Visser

The use of an Ordered Binary Decision Diagram (OBDD) to store all visited states during on-the-fly model checking (or reachability analysis) is investigated. To improve the time and space efficiency a novel state compression technique is introduced. This compression technique is safe, in the sense that no two unique states will have the same compressed representation. A number of real-world (as opposed to contrived) examples are used to evaluate an experimental implementation of the OBDD state store within the SPIN validation tool. In all the examples a reduction in space is achieved when using the OBDD state store as opposed to the more traditional hash table state store. The memory and time usage when combining partial orders with the OBDD state store is also considered.