REPORTS ON CONFERENCES

Report on BCTCS 13 at the University of Sheffield, 1997

The 13th Annual Meeting of the British Colloquium for Theoretical Computer Science was hosted by the University of Sheffield between the 23rd and 26th March 1997. The colloquium was held in the pleasant surroundings of Halifax Hall, located on the outskirts of Sheffield, one of the major cities of the North of England.

There was an interesting and wide-ranging programme of invited talks and tutorial presentations. Among these, special mention should made of the invited talk given by Trevor Bench-Capon who at extremely short notice, stood in for Leslie Goldberg who was indisposed, presenting a thought-provoking personal view on the limits of Artificial Intelligence. Other invited presentations were given by: Martyn Amos (Liverpool); Paul Dunne (Liverpool); Javier Esparza (Munich); Matt Fairtlough (Sheffield); Mike Gordon (Cambridge); Edmund Robertson (St. Andrew's); Edmund Robinson (QMW, London); and Chris Tofts (Leeds). The invited programme was well complemented by a total of 31 contributed talks.

It would be difficult to find fault with the skillful organisation of the meeting by Matt Fairtlough and Mike Holcombe, and the Colloquium continues to be an important event in the British Theoretical Computer Science community.

BCTCS 14, the running of which is in the capable hands of Tom Kelsey and Prof. Ursula Martin, will be held at the University of St. Andrew's in Scotland from March 31st to April 2nd, 1998. Anyone wishing to contribute talks concerning Theoretical Computer Science, is warmly encouraged to do so. Further information regarding BCTCS 14 may be found on the Web at

http://www-theory.dcs.st-and.ac.uk/events/bctcs14.html.

The BCTCS Web Pages, giving details concerning previous colloquia, invited presentations, and the text of the BCTCS Constitution adopted at the Sheffield meeting are located at **http://www.csc.liv.ac.uk/~ped/bctcs/summary.html**.

Paul E. Dunne (Liverpool)
Matt Fairtlough (Sheffield)

BCTCS 13 23-26th March 1997
Abstracts of invited talks and tutorials

Martyn Amos
Dept. of Computer Science, University of Liverpool

The Complexity and Viability of DNA Computations

The idea that living cells and molecular complexes can be viewed as potential machinic components dates back to the late 1950s, when Richard Feynman delivered his famous paper describing sub-micro-scopic computers. Recently, several papers have advocated the realisation of massively parallel computation using the techniques and chemistry of molecular biology. Algorithms are not executed on a traditional, silicon-based computer, but instead employ the test-tube technology of genetic engineering. By representing information as sequences of bases in DNA molecules, existing DNA-manipulation techniques may be used to quickly detect and amplify desirable solutions to a given problem.

In this talk we provide an introduction to DNA computation, focusing in particular on complexity issues. We believe that these issues are paramount in the search for so-called "killer applications", that is, applications of DNA computation that would establish the superiority of this paradigm over others in particular domains. An assured future for DNA computation can only be established through the discovery of such applications.

Trevor Bench-Capon
Dept. of Computer Science, University of Liverpool

Machines Can't Think

For many the avowed aim of Artificial Intelligence is to produce a machine that thinks. But is it really possible to say that a machine thinks? This is not an empirical question, which can be settled by observing behaviour, but relates to how we (the linguistic community) choose to describe behaviour. There is a metaphorical use which ascribes intensional concepts to machines to provide pragmatically helpful explanations, but this is only a manner of speaking not a serious claim that computers have minds. To claim that machines think is to ascribe them a status traditionally reserved for persons, and to take a particular social and ethical stance towards them. At present this is inappropriate. Machines will only think if we change our view of people, so that we see them - and ourselves - as machines.

Paul E. Dunne
Dept. of Computer Science, University of Liverpool

An Overview of Lower Bound Techniques in Monotone Boolean Function Complexity

In 1985, Razborov discovered the 'Method of Approximations', an approach that led to the first super-polynomial lower bounds on the monotone circuit size of explicitly defined Boolean functions. Over the last few years there have been further advances in the theory of monotone circuit complexity, resulting in extensions to Razborov's approach and the discovery of new lower bound arguments, e.g. Haken's 'bottleneck counting' technique, Finite Limits, etc. In this talk, an overview and comparison of these approaches will be presented. The potential for obtaining lower bounds on general Boolean networks will also be considered.

Javier Esparza
Technical University of Munich

Model-Checking Pushdown Automata

Pushdown automata (PDA) can be seen not only as language acceptors, but also as processes. As a process, the semantic of a PDA is no longer a language, but a - possibly infinite - Kripke structure (or transition system). Temporal logics can be used as query languages to describe properties of this structure. The model-checking problem consists of deciding whether the Kripke structure of a given PDA is model of a given temporal formula or not. Since the Kripke structure associated to a PDA can be infinite, the many existing model-checking techniques for finite structures cannot be applied. Several papers have proposed solutions to the model-checking problem for the modal mu-calculus, a particularly powerful temporal logic, or fragments thereof (Burkart und Steffen 92 und 94, Hungar und Steffen 93, Walukiewicz 96). In the talk I present algorithms for linear and branching-time logics, which I claim are easier to understand than the existing ones. These algorithms rely on a simple result of automata theory which is part of my

undergraduate lecture on formal languages. This is joint work with Ahmed Bouajjani and Oded Maler, Verimag (Grenoble).

Matt Fairtlough
University of Sheffield

Abstraction, constraints and the λ-calculus

This talk will review the "Propositions as Types" correspondence between constructive logic and the λ-calculus, through to a first-order extension of Moggi's computational λ-calculus, corresponding to a logic which we shall call "Quantified Lax Logic", or QLL. Some concrete examples from hardware and software verification will be indicated.

Virtually all scientific investigations introduce some form of abstraction to deal with the enormous complexity presented to us by the world. The constraints under which the abstraction is valid are not always explicitly acknowledged or even understood, although they must be always be present in some form. An obvious example is Newtonian mechanics, where the abstract model of behaviour is accurate provided the speeds and gravitational fields involved are sufficiently small. The model may be refined to a more accurate relativistic form, which would itself embody a refined notion of constraint. Similar hierarchical chains of abstractions frequently arise in hard- or software verification. In situations where the relevant constraints have been identified and system behaviour can be adequately formalised in logic, it seems worthwhile to try to provide a general theory and a formal mechanism for handling the awkward passage across abstraction boundaries - indeed it may well be that most errors in system verification creep in at these boundaries.

I will show how QLL may be used to give a theory of the abstraction process and to automate the process of constraint extraction from abstract correctness proofs.

Mike Gordon
University of Cambridge

Event and cycle semantics of hardware description languages

Modern hardware description languages (HDLs) are interpreted in different ways for different purposes. Classical simulation uses an event semantics to model detailed asynchronous behaviour. However, current compilers from HDLs to circuits generally target clocked synchronous implementations and so use a synchronous cycle-based semantics, as do high speed cycle simulators. The talk will start with a tutorial on event and cycle semantics (aimed at computer scientists, not electrical engineers) and then outline current work at Cambridge on developing semantics for Verilog, the most widely used HDL (62 percent market share).

Edmund Robertson
University of St. Andrew's

Combinatorial and decidability questions in semigroups of words

Let A be a finite alphabet and let $F(A)$ be the set of all words in A. $F(A)$ becomes a semigroup by defining multiplication as concatenation of words. $F(A)$ is called the free semigroup on A. Given a finite subset B of $F(A)$, let T be the smallest subset of $F(A)$ containing B which is closed under multiplication. We show that T need not be a free semigroup and examine questions relating to how close T is to being free. A general T may be free or may not be finitely presented. By way of contrast, if T is an ideal of S then T is never free but is always finitely presented.

Edmund Robinson
Queen Mary and Westfield College, London

Logic and Logical Relations

Logical relations have proved a fertile ground for generalisations, mostly fairly ad hoc. These generalisations have in turn inspired attempts to furnish a general formulation of the constructions. Most of these have been based on variants of the categorical theory of relations. Work by Hermida, however, makes precise an interesting connection with formal logical systems. This has the advantage of connecting the logical relations directly with systems for reasoning about the types involved. This talk will discuss that connection, and some more recent developments arising from it.

Chris Tofts
School of Computer Studies, Leeds University

Ants, Shrimps and Other Asynchronous Hardware

Process Algebra has been used as a method for describing concurrent computing systems for some time. Extensions to the underlying framework that permit the description of time, probabilistic and priority effects have also been explored. In this talk I will present how the process algebraic approach can be exploited to model the behaviour of complex systems. In the Biological setting it permits us to achieve formal individual based models. I will present models of task allocation behaviour in ants, and the evolution of sex determination strategies in shrimp. These same methods can be applied to performance analysis in concurrent systems in particular that of asynchronous hardware.

Abstracts of contributed talks

Andrew Adams
Department of Computer Science, University of St. Andrews

Formal Representation of Sequent-Style Calculi

Formalisation of meta-theoretic proofs about logics, and in particular about sequent-style calculi, is becoming more common. There are still few tools around which support such formalisations with any degree of specialisation. Each researcher must produce their own formalisation of the syntax. I will describe a number of approaches taken in the last five years, together with some commentary on the pros and cons of each style.

Yamine Ait-Ameur
LISA/ENSMA, Site du Futuroscope

Representation of abstract interpretations of functional languages in Kripke semantics

Many efforts have been devoted in the area of building formal proof systems for the development and the verification of software. Several of these systems are based on the notion of types in programming languages and on the correspondence between types in programming and propositions in logic. These systems have been used for developing programs from a specification and moreover for program verification.

Verification techniques were used to check if a piece of software satisfies its specifications or a certain property. Logics, transformation systems or formal methods have been designed for this purpose.

The origin of the work we intend to present in this paper is double.

On the one hand, other techniques for the verification of program properties have been developed and among them the abstract interpretation technique. It allows to infer properties of a given object by running a correct abstract version of this object on an abstract domain.

On the other hand, Kripke semantics allows us to represent the semantics of a given object in different worlds and Kripke applicative structures give the semantics of typed functional languages and of types. It gives us a common structure allowing to reason on objects. We use, in this paper, these structures to reason on the standard and non standard semantics of programs. In this paper, by non-standard semantics, we mean the semantics obtained by abstract interpretation.

The goal of our work is to show that it is possible to formally consider different semantics of a given program in a common structure. This paper gives a representation of standard and non standard semantics in general and abstract interpretation in particular in Kripke semantics using applicative structures. It expresses that it is possible to check the correctness of abstract interpretations by type checking. The idea is to change the world where the program is defined by an abstract world.

In order to achieve this goal we need:

1. to define a relation between the two worlds: the concrete and the abstract worlds,
2. to set the safety conditions which ensure the correctness.

The approach we have followed is the one described by S. Absramsky. The solution consists in defining a logical relation to link the typed objects of the two worlds. This relation allows to encode the semi-homomorphism property of the abstraction function. Then, the safety condition is stated by defining a condition on the defined logical relation. At this level, it is possible to establish the theorem which states formally the safety of the defined abstract interpretation using the defined logical relation.

The equations and the proofs performed in this paper have been implemented in the DEVA type system which allows to check proofs thanks to the "propositions as types" paradigm. Moreover, two examples : the rule of signs and strictness analysis are studied within the developed approach.

Chris Angus
Department of Computer Science, Newcastle University

Using Monad Transformers to debug under program transformation

Many functional language compilers such as GHC carry out a great proportion of their work by means of correctness-preserving, and hopefully performance improving, program transformations. This approach is attractive since each transformation can be implemented, verified and tested in isolation, leading to fewer monolithic passes.

The application of these transformations however, leads to code which bears little resemblance to original source code, and therefore, any attempt to relate information gained from the evaluation of transformed expressions to source code, for debugging purposes, is difficult.

We show how monad transformers may be used to structure these transformations so information necessary to relate transformed expressions to source expressions may be generated and recorded by interpreting our transformations under the monad formed by

applying our monad transformer to an information gathering monad. This approach has the following advantages:

- Debugging may be turned Off or On by applying the monad transformer to the identity monad or an information gathering monad respectively.
- Each transformation may be verified and checked as before by applying the monad transformer to the identity monad.
- The process of proving transformations correct under an information gathering monad is simplified to showing this monad is homomorphic to the identity monad.

We conclude by sketching how this information may be used in the context of a functional language debugger.

Yaagoub A. Ashir and *Iain A. Stewart*
Department of Mathematics and Computer Science, Leicester University

Broadcast and Scatter Communication Algorithms in k-ary n-cube Interconnection Networks

The k-ary n-cube interconnection networks are cubes with n dimensions and k processors in each dimension. A k-ary n-cube parallel processor consists of k^n identical processors, each provided with its own sizable memory and interconnected with $2n$ neighbors. The k-ary n-cube has some attractive features like symmetries, high level of concurrency and efficiency, regularity and high potential for the parallel execution of various algorithms. The k-ary n-cube network has smaller diameter than the mesh and lower node degree than the hypercube. In this talk we present the recursive structure of the k-ary n-cube network and some of the topological properties of this model. To extend the strategy of the binary Gray codes, we introduce a class of generalized Gray codes called k-ary Gray codes and show the recursive structure of the Hamiltonian cycle. Then using the Gray codes strategy, we develop and analyze the broadcast and scatter communication algorithms.

The multi-node broadcast, when every processor wishes to send a message to every other processor, and the single-node scatter, when a source processor wishes to send a different message to every other processor, can also be developed using the node-disjoint cycles of the k-ary n-cube model which utilizes the more efficient dimensional routing scheme in each phase of the algorithms. All of our algorithms are optimal under the assumption of one-port I/O communications and store-and-forward routing.

Mike Holcombe and *Kirill Bogdanov*
Department of Computer Science, Sheffield University

The third step towards correct software

At the moment most blackbox testing methods are empirical and aim at finding faults rather than assuring correctness. Our target is to develop a testing method allowing one to prove software to be correctly implemented, up to some clearly defined constraints, as a result of non-exhaustive testing, where the system is specified in a suitable formal notation.

The first step towards this goal was carried out by Chow, 78 for finite-state machine specifications. It was extended to cope with the more general case of X-machines by Ipate, 95.

The authors try to develop it further to deal with systems specified in the language, statecharts, which is used extensively in industry to specify reactive systems. The approach taken involves the formalisation of the statechart notation, its extension through the

incorporation of a formal data modelling component (based on Z) and the extension of the testing method developed for X-machines to deal with various features of these statecharts. The talk will present the mapping of statecharts to X-machines.

This project is sponsored by Daimler-Benz, Research and Technology, Berlin.

Alexander Bolotov
Department of Computing, Manchester Metropolitan University

Clausal Temporal Resolution in a Branching-Time Framework

While much of the research into the verification of concurrent and distributed systems using branching-time temporal logics has centred around the model-checking technique, relatively little research has been carried out on efficient decision procedures for such logics, such as the resolution based approaches. Recently, however, several applications requiring refined proof methods for branching-time temporal logics have appeared, most notably the specification and verification of multi-agent systems. We are currently extending a clausal resolution-based proof method for linear-time temporal logic developed by Fisher. This involves extending its key elements, namely the concept of the normal form and two types of resolution rule (classical and temporal), to the branching-time context (in particular, the CTL family of logics). The temporal resolution here should allow us to resolve constraints expressing "*always A*" and "*sometime not-A*" when they refer to the same path. We define a normal form for each branching-time logic considered and present rules to transform formulae into the normal form which are built upon the fixpoint characterisation of the basic modalities. The most difficult case of CTL^* also involves a special technique to express "path contexts" of formulae in the normal form. Soundness and completeness arguments are provided as well as some prospects of future research.

Russ Bubley and *Martin Dyer*
Department of Computer Science, Leeds University

Improved Random Generation of Linear Extensions

We examine the problem of sampling (almost) uniformly from the set of linear extensions of a partial order, a classic problem in the theory of approximate sampling. Previous techniques have relied on deep geometric arguments, or have not worked in full generality. Recently, focus has centred on the Karzanov and Khachiyan Markov chain. In this paper, we present a significantly simpler proof of the rapid mixing of this chain using path coupling. We further show that the mixing time is lower than was previously known.

Virgil Emil Cazanescu
University of Bucharest

The algebra of programs

We introduce an algebraic structure (biflow) based on operations called composition, (separated) sum and feedback. Biflows are used to study the syntax and semantics of deterministic and nondeterministic programs (sequential processes). The biflow of programs built using statements from a set X together with a connection from a biflow B is the coproduct of B with the biflow freely generated from X. This research is an extension of work by C.C. Elgot.

Athina Christodoulou
Department of Computer Science, Leeds University

Modelling spawning processes in WSCCS

WSCCS is a probabilistic calculus, derived from Milner's SCCS, by adding a probabilistic quantification to non-deterministic choice operations. The Process Algebra Toolset, accepts a restricted form of WSCCS expressions, and builds their probabilistic transition graphs. The tool can check for deadlocks and livelocks, and also generate algebraic expressions for the calculation of probability (or the mean time taken) of observing particular actions.

WSCCS can describe processes having infinite state spaces. Unfortunately infinite state processes cannot be handled directly by finite computing system. A special class of infinite state processes are spawners; processes which can create other processes. I discuss a way to generate approximate transition graphs for spawning processes which cover a finite subset of the infinite state space. In the case of probabilistic systems, a finite approximation can cover a large amount of the probabilistic behaviour of the complete system. System properties can then be derived automatically on these finite approximations to the complete system.

Mary Cryan, *Leslie Goldberg* and *Cynthia Phillips*
Warwick University

Approximation Algorithms for the Fixed-Topology Phylogenetic Number Problem

In evolutionary biology the characteristics of a set of species S are often modeled by a set of characteristic functions of the form $c : S \rightarrow R_c$, where R_c is the set of the different states of the characteristic c. Given a set of species S and its characteristic functions C, a phylogeny is a tree T whose leaves are labeled by species from S and whose internal nodes are labeled by species which can be defined by extending the functions in C. A hypothetical evolutionary history for a set of species S with characteristic functions C is obtained by constructing some optimal phylogeny.

Optimal character-based phylogenies usually minimize the number of connected components in T formed by the states of the characteristic functions. Consider the Fixed-Topology L-phylogeny problem: given a set of species S and its characteristic functions, and a tree T whose leaves are labeled by species from S, can we label the internal nodes of T so that the number of connected components in T for any state of any character is at most L? The Fixed-Topology Phylogenetic Number Problem is the problem of finding the minimum L for which the input has an L-phylogeny. Both of these problems are NP-hard (for $L>2$). In my talk I will present a 2-approximation algorithm for the Fixed-Topology Phylogenetic Number Problem and I will also present an algorithm which finds a 4-phylogeny for any fixed-topology input which has a 3-phylogeny.

Chris Dornan
Department of Computer Science, University College, Cork

Affordable Dynamic Types

This paper considers a new approach to embedding dynamic typing facilities in statically-typed functional languages like ML and Haskell. It is substantially simpler then existing proposals for polymorphic types which, as their authors admit, are rather complicated [1,§7] [4,§6]. This proposal is based on a simple tag comparison scheme that takes account of polytypes.

In fact, this proposal recycles an old idea from functional languages with an exclusively dynamic type-discipline in which atomic type-testing predicate functions are used to test

for objects like integers, characters and list cells. It works well because of the use of automatic storage management schemes that pretty-much force the heap to be tagged; it is easy to add the run-time type tags to this scheme so that each tag is shared on a per-type basis.

Essentially the same technique is being proposed here for statically typed languages, where each data type declaration would generate an extra pattern constructor for testing for values of the data type with a simple tag test.

From the type-checking point of view, input expressions may be of any type, but the result of a successful match is the type being tested for. Parametric types, when matched, will generate existential types [5,3].

In order to properly support polytypes, data-type declarations are extended with local type variables that can be either existentially bound [3] or universally-bound [2]. Existentially bound variables allow values of arbitrary type to be encapsulated in data types while universally-bound variables provide for the transmission of polymorphic values.

Though relatively simple, this scheme loses little in practical terms over the more complicated proposals [2].

Bibliography

[1] M.Abadi, L.Cardelli, B.Pierce, and D.Remy. Dynamic typing in polymorphic languages. Journal of Functional Programming}, 5(1):111-130, January 1995.

[2] C.B.Dornan. Type-Secure Meta-Programming. PhD thesis, University of Bristol, 1997.In preparation.

[3] K.Laufer and M.Odersky. Polymorphic type inference and abstract data types. Transactions On Programming Languages and Systems, 16(5):1411-1430, September 1994.

[4] X.Leroy and M.Mauny. Dynamics in ML. Journal of Functional Programming, 3(4):431-463, October 1993.

[5] N.Perry. The Implementation of Practical Functional Programming Languages. PhD thesis, Imperial College, London, 1991. Second edition.

Billy Duckworth and *Alan Gibbons*
Dept. of Computer Science, Liverpool University

Simpler proofs for Hypercubic Permutation Routing and Hamiltonian Decomposition

The importance of the hypercube as an interconnection network for parallel computers has stimulated much research into its structural properties and its ability to support efficient parallel algorithmics. A great deal is now known about the hypercube in these regards. For pedagogic purposes, several proofs in the literature are not as elegant or as simple as they might be. In this paper we present much simpler proofs, not to be found elsewhere, of two well-known facts about hypercubes: that for the d-dimensional hypercube there exists sets of paths by which any permutation routing task may be accomplished in at most $(2d-1)$ steps without queuing and, for even d, there exists a decomposition of the hypercube into $d/2$ edge-disjoint Hamiltonian cycles.

James Dyer
School of Computer Studies, Leeds University

A Compositional Approach to Telecommunications Network Performance Modelling

At the present, analysis of large systems is limited due to state explosion. However, in the case of probabilistic systems we may approximate by a simpler system.

This talk aims to illustrate this by means of an example. We provide an analysis of the DS-CDMA-S-ALOHA (Direct-Sequence Code-Division-Multiple-Access with slotted ALOHA random access) protocol. The analysis is corroborated by results from a discrete Markov Chain analysis given in an earlier work.

The protocol has been modelled in WSCCS, a probabilistic variant of SCCS. This notation provides an easily readable syntactic description of the problem. This is then approximated to a simpler model. This subsequent model has provided us with symbolic solutions for stability and expected delay.

This project is supported by an EPSRC CASE award sponsored by British Telecom.

Russ Bubley and *Martin Dyer*
Department of Computer Science, Leeds University

Approximate counting and path coupling

The method of coupling for proving rapid mixing of certain Markov chains, which occur in applications to approximate counting, will be reviewed briefly. A new idea, which makes this somewhat easier to apply, will be outlined and illustrated.

Willem Fokkink
Department of Mathematics and Computer Science, University College of Swansea

Iteration or Recursion, a Difficult Choice

In 1956 Kleene introduced the iteration operator, which makes a basis for formal language theory. In 1966 Salomaa gave an axiomatization which completely characterizes iteration in trace semantics. When Milner introduced process algebra, he also considered iteration as a means to describe infinite processes, and adapted Salomaa's axiomatization to characterize iteration in bisimulation semantics. Milner asked whether his axiomatization is complete. This question, which was raised in 1984, remains unsolved. Recently some results have been proved for subalgebras of process algebra with iteration that might lead to a positive answer to Milner's question.

Richard Gault
Department of Mathematics and Computer Science, Leicester University

Playing games with transitive closure

The question of whether or not $NP=co\text{-}NP$ is one of the most fundamental open problems in theoretical computer science. In a seminal result in the early 1970's, Fagin characterised *NP* as the class of problems expressible using second order existential logic, and went on to show that an important subclass of this — monadic NP — is, in fact, not closed under complementation.

We extend monadic *NP* by adding a transitive closure operator, and investigate the resulting logic. In particular, we show that it too has an important subclass which is not closed under complementation.

Ali Hamie
School of Computing and Mathematical Sciences, University of Brighton

A Compositional Semantics of Object Diagrams

We describe a compositional formal semantics of object model and statechart diagrams as used in the Syntropy method of Object-Oriented Analysis and Design. Separate theories are constructed for each element in the diagram, i.e. object types, associations, and subtyping which are then combined through inclusions to yield a formal semantics of the complete system. The semantics is expressed in the Larch Shared Language (LSL).

In interpreting the statecharts, we consider statecharts as partitioning the overall statespace and defining transitions between partitions. We also provide a formal semantics to event parameters, filters, preconditions, and postconditions.

Rob M. Hierons
Department of Mathematical and Computing Sciences, Goldsmiths, University of London

Testing from a Finite State Machine using Overlap

In order to test against a FSM is necessary to check the transitions. Testing a transition involves executing a section in the form of a transition followed by a state identification sequence: these will be called test subsequences. Aho et al. [1988] express the problem of finding the shortest sequence as that of minimally connecting these test subsequences.

Yang and Ural [1990] note that the test subsequences may overlap. They utilize a form of overlap, in which one test subsequence $t1$ is in the form of a transition followed by (the beginning of) another test subsequence $t2$. Unfortunately, while their method produces shorter test sequences, it relies upon heuristics and so can be suboptimal.

Hierons [1996] proves that the form of overlap utilized by Yang and Ural can be fully represented using invertible transitions. The problem of producing the shortest such test sequence can be represented as a network optimization algorithm.

Overlap can be extended from transitions to sequences in a natural way. This allows general test overlap to be fully represented. A network optimization algorithm, using invertible sequences, can be produced to find the shortest test sequence that allows any form of overlap.

It transpires that invertible sequences have a number of nice properties and relate to state identification sequences. It is thus possible, during the process of finding invertible sequences, to generate the state identification sequences.

References:

A.V. Aho, A.T. Dahbura, D. Lee, and M.U. Uyar, 1988. An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, Proceedings of Protocol Specification, Testing, and Verification VIII, pp75-86, Atlantic City, North-Holland.

Hierons R.M. 1996. Extending Test Sequence Overlap by Invertibility, The Computer Journal Vol 39 No 4, pp 325-330.

B. Yang and H. Ural, 1990, Protocol Conformance Test Generation Using Multiple UIO Sequences with Overlapping, ACM SIGCOMM 90: Communications, Architectures, and Protocols, Sept 24-27 p118-125, Twente, Netherlands, North-Holland.

Florentin Ipate and *Mike Holcombe*
Department of Computer Science, Sheffield University

A method for refining and testing generalised machine specifications

Two current areas of emphasis in the development of methods for engineering higher quality and safer software are the use of formal methods for the specification and verification of software and the development of sophisticated methods of software and system testing. In general these two approaches are unrelated and few examples exist whereby the best aspects of both are used in an integrated way. Ipate and Holcombe present such an integrated approach by using a type of generalised machine, namely the stream X-machine, both as a specification tool and as a basis of a testing method. However, if this type of specification method is to become usable in a wide range of software applications and acceptable to a wide community of software engineers then there needs to be ways of refining existing specifications into more complex and more detailed versions. Also, for each such process of refinement, there needs to be methods of deriving the test set for the resulting machine from that of the initial machine. This paper present such a refinement and also provides a method for testing machines constructed through such a process of refinement.

Ranko Lazic and *Bill Roscoe*
Oxford University Computing Laboratory

Data Independence

A process can be said to be data-independent with respect to a data type T if it can communicate (i.e. input from its environment and output back to its environment) values of type T, but not perform any computations on them. An example would be a buffer process: a buffer of some type T can indeed receive, store and send values of type T, but it never performs any computations on them. Many algorithms and systems (especially when sufficiently abstracted) are data-independent.

When a process is data-independent with respect to T, it makes sense to substitute any concrete data type for T. Indeed, data-independence can be thought of as the same thing as polymorphism; and the basic theory of logical relations in functional programming can be adapted to processes, and used to reason about semantics of data-independent processes.

If a specification Spec and a proposed implementation Impl are both data-independent with respect to a type T, there are infinitely many correctness questions:

$Q(n)$: Is Spec satisfied by Impl when T has n elements?

Fortunately, it often turns out that, depending on Spec and Impl, we can calculate N such that it suffices to consider $Q(N)$! More precisely, $Q(N)$ is going to be equivalent to any $Q(n)$ for $n \geq N$. Theorems which tell us when and how we can compute such N are obviously very useful for automated verification.

Work on data-independence can also be applied to verifying processes which contain n parallel components, where the components are typically identical and n can vary arbitrarily.

Ranko Lazic and *Bill Roscoe*
Oxford University Computing Laboratory

Canonical Transition Systems and Non-well-founded Sets

Transition systems (i.e. directed graphs with labelled arcs) are common in theoretical computer science, chiefly as vehicles for operational semantics.

Given a point in a transition system, its "n-approximation" records the first n steps of its behaviour. Generalising this idea, we can define α-approximations for all ordinal numbers α. Two points in a transition system are then strongly bisimilar if and only if, for any α, their α-approximations are the same.

The notion of α-approximation gives rise to a hierarchy of canonical transition systems, where the canonical transition system at level α consists of all possible α-approximations. The hierarchy has very rich combinatorial and topological structure. It also admits some useful fixed point theorems.

This kind of study of transition systems is very closely connected to the theory of non-well-founded sets. (The simplest example of a non-well-founded set is a set whose only member is itself.) Indeed, the hierarchy of canonical transition systems gives rise to a model of set theory in which the Axiom of Foundation is replaced by Aczel's Anti-Foundation Axiom, which postulates the existence of many non-well-founded sets. The model obtained in this way has very interesting structure, and it can be used to prove some results about consistency and independence in the presence of the Anti-Foundation Axiom.

Juha Nurmonen
Department of Mathematics and Computer Science, Leicester University

About locality of first-order logic and some extension

Many important tools in model theory, such as compactness and completeness theorems, fail when restricted to finite models. When studying limitations of expressive power of logics, typically one has to use Ehrenfeucht-Fraisse games. Such a game theoretic approach is known for practically every logics studied in finite model theory.

To show limitations of expressive power of first-order logic several proposals have been suggested to replace the game theoretic approach by a combinatorial argument. Each of these arguments formalize in a sense a notion of locality. Several formulations have been given to show that a first-order query looks at small portions of input, but cannot tell about the whole structure: Gaifman (1982) showed that every first-order formula is equivalent to a local one in a sense that only a small part of the input is relevant to evaluate the query. Hanf (1965) introduced a method (for general model theory) based on counting the numbers of small neighborhoods in structures, and this method was modified for finite model theory by Fagin, Stockmeyer and Vardi (1995). Libkin and Wong (1994) proved the bounded degree property of first-order queries, which intuitively gives another way to say that if locally input looks simple, then so does the output.

We look at these notions of locality and give some examples of limitations of expressive power of first-order logic. We also consider some extensions of first-order logic and see how such notions can be used for showing limitations of expressive power of such extensions.

Wojciech Rytter
Dept. of Computer Science, Liverpool University

Efficient Algorithms for Compressed One- and Two-Dimensional Texts

We consider several basic problems for texts and show that if the input texts are given by their Lempel-Ziv codes then the problems can be solved deterministically in polynomial time in the case when the original (uncompressed) texts are of exponential size. The growing importance of massively stored information requires new approaches to algorithms for compressed texts without decompressing.

Denote by $LZ(w)$ the version of a string w produced by Lempel-Ziv encoding algorithm. We consider classical algorithmic problems for texts in the compressed setting}. For given compressed strings $LZ(T)$, $LZ(P)$ we give the first known polynomial time algorithms to compute compressed representations of the set of

1. all occurrences of the pattern P in T,
2. all periods of T,
3. all palindromes of T,
4. and all squares of T.

Then we consider several classical language recognition problems in the compressed setting.

We consider also the complexity of problems related to 2-dimensional texts (2d-texts) described succinctly. In a succinct description, larger rectangular sub-texts are defined in terms of smaller parts in a way similar to that of Lempel-Ziv compression for 1-dimensional texts. A given 2d-text T with many internal repetitions can have a hierarchical description which is up to exponentially smaller and which can be the only part of the input for a pattern-matching algorithm which gives information about T. Such a hierarchical description is given in terms of a straight-line program, or, equivalently, a 2-dimensional grammar. It also resembles hierarchical descriptions of graphs. We show that the complexity dramatically increases in a 2-dimensional setting (compared with 1-dimensional case).

This is a joint work with L.Gasieniec, M.Karpinski, and W.Plandowski.

Tony Simons
Department of Computer Science, Sheffield University

A Theory of Class

We present a mathematical theory of class. The motivation for this theory is initially to explain the operational behaviour of objects in object-oriented languages. However, the theory is truly general, in that it encompasses many different approaches to type abstraction, such as type constructors, generic parameters, classes, inheritance and polymorphism. The theory is elegant, in that it is based on a simple generalisation of Cook et al.'s model of F-bounded polymorphism. Examples involving the construction of objects, types, classes, inheritance, type constructors and generic classes will be presented. The theory has implications for future language designs, since it unifies the different mechanisms for polymorphism in current object-oriented languages. It also has timely implications for emerging OMG standards defining terms such as 'class', 'type' and 'interface'.

Jason Steggles
Department of Computer Science, Newcastle University

Parameterised Higher-Order Algebraic Specifications

Higher-order algebraic methods provide a natural and expressive formal framework for specifying and reasoning about computing systems. However, further work is needed to

extend these methods into a practical formal approach for system design. In this talk we begin to address this issue by developing a semantics for parameterised higher-order algebraic specifications. By a parameterised specification we mean a specification with a formal parameter part the structure and semantics of which are left open. Thus parameterised specifications allow generic data types to be specified and facilitate the reuse of specifications. We begin with the well-known free functor semantics for first-order parameterised specifications but observe that due to the nature of the higher-order initial model these results cannot be extended to the higher-order case. We then consider a concrete construction of a functor which we propose as the semantics of a parameterised higher-order specification. We demonstrate the theory we develop by constructing a parameterised second-order equational specification for convolution.

Iain Stewart
Department of Mathematics and Computer Science, Leicester University

Fault-tolerant embeddings of Hamiltonian circuits in k-ary n-cubes

We consider the fault-tolerant capabilities of networks of processors whose underlying topology is that of the k-ary n-cube Q_n^k, where $k \geq 3$ and $n \geq 2$. In particular, given a copy of Q_n^k where some of the inter-processor links may be faulty but where every processor is incident with at least two healthy links, we show that if the number of faults is at most $4n-5$ then Q_n^k still contains a Hamiltonian circuit, but that there are situations where the number of faults is $4n-4$ (and every processor is incident with at least two healthy links) and no Hamiltonian circuit exists. We also show that given a faulty Q_n^k, the problem of deciding whether there exists a Hamiltonian circuit is NP-complete.

Alastair Telford
University of Kent at Canterbury

Codata and Corecursion

We wish to design an elementary strong functional language. It should have type theory's property of strong normalisation whilst being elementary like Miranda or Haskell in that the type system should be (a simple variant of) Hindley-Milner. Such a language should include schemes for well-founded recursion which are not overly restrictive in terms of expressive power. We would also require in our language the ability to perform operations upon infinite data structures.

In this talk we show how a scheme of corecursion over codata (infinite structures) may be formulated for such a language. In particular, we show how both strong normalisation and the Church-Rosser property can be ensured for functions over infinite data structures.

Rick Thomas
Department of Mathematics and Computer Science, Leicester University

Automatic Semigroups

This talk describes some joint work with Colin Campbell, Edmund Robertson and Nik Ruskuc from St Andrews. The notion of an automatic group is now an established one in group theory and there have been a variety of interesting results proved about them. Roughly speaking, a group is "automatic" if, firstly, there is a normal form for its elements that can be recognized by a finite automaton, and, secondly, the results of multiplying elements in the group by generators can also be recognized by finite automata. The purpose of this talk is to describe how one can generalize this notion from groups to arbitrary semigroups. In particular, we shall highlight some results from the group theory

setting which still hold in this more general context, but we will also mention some areas where automatic semigroups can behave differently to automatic groups.

John Tucker
Department of Mathematics and Computer Science, University College of Swansea

Computability theory for topological data types

There is renewed interest in computability theories on the real numbers and other continuous algebras following the popularisation of the topic by Blum, Shub and Smale in Bull AMS in 1989. Of course the subject is quite old and has many interesting problems and results.

In this talk I will review and classify various models of computation in the general setting of continuous many sorted algebras including:

1. Pour El and Richards axiomatic approach to computable analysis.
2. Weirauch's approach to computable analysis.
3. Domain representability of data types.
4. Imperative models and other generalisations of classical computability models.

I will report on new results in these areas, mainly from my collaboration with V Stoltenberg-Hansen (Uppsala) and J I Zucker (McMaster).

Matt Walton
Department of Computer Science, Sheffield University

A Lax Logic View of Constraint Logic Programming

Constraint Logic Programming (CLP) is the result of a merger between two declarative paradigms - constraint solving and logic programming. It has been successfully used in many applications, from scheduling to the synthesis of analogue circuits. Various semantics and proof-theoretic characterisations have been proposed for CLP but we offer a novel way of 'factoring out' the constraints from the logic in both cases.

Lax Logic is an intuitionistic modal logic, used originally in the formal verification of hardware. It also provides, however, new semantic and proof-theoretic perspectives on CLP. As program clauses in pure logic programs correspond to the Horn-clause fragment of first-order intuitionistic logic, so program clauses in CLP programs can be written in the Horn-clause fragment of first-order Lax Logic. The difference with the latter is that the constraints are abstracted away using a single modal operator, leaving behind a (Lax) logical structure. From this we have a purely logical proof-theoretic characterisation.

Further, by the Curry-Howard isomorphism, natural deduction proofs in Lax Logic give rise to terms of the computational lambda calculus. Using a particular notion of computation, such terms can recover concrete answer constraints for the CLP program from Lax Logic derivations.

Lax Logic thus provides mechanisms for abstraction and constraint extraction, with possible applications to the refinement and 'abstract debugging' of logic programs.

Paul E. Dunne and *Michele Zito*
Dept. of Computer Science, Liverpool University

On the 3-Colourability Threshold

Experimental studies indicate that a number of NP-complete decision problems Π exhibit the following behaviour: if X_n is the set of instances of size n and m is some parameter of

instances $x \in X_n$, then instances for which m is small are almost all such that $x \in \Pi$ and instances for which m is large almost all are such that $x \notin \Pi$. Furthermore there is an apparent threshold m^* such that $m<m^*$ implies that almost all x belong to Π and $m>m^*$ implies that almost all x do not belong to Π. We obtain upper bounds on the threshold in 3-colouring n-vertex graphs. An elementary first-moment method suffices to show that almost all graphs with average degree at least 5.419 are not 3-colourable. An improved analysis brings this constant down to 5.277. The technique generalizes to k-colouring.

Report on 2nd ERCIM International Workshop on Formal Methods for Industrial Critical Systems (FMICS'97) July 4–5 1997, Cesena (I)

The second ERCIM International Workshop on Formal Methods for Industrial Critical Systems took pla at University of Bologna, located in Cesena, on July 4-5, 1997.

The workshop has been organized as a satellite meeting of the 24th International Colloquium on Automat Languages and Programming - ICALP 97 and was the second meeting of the ERCIM Working Group on Form Methods for Industrial Critical Systems - FMICS since its inception, in 1996.

About 30 people attended the workshop, which was chaired by S. Gnesi of CNR/IEI and locally organiz by Roberto Gorrieri (Univ. of Bologna). A total of 16 presentations were given on several different aspects application of formal methods. The following four invited speakers gave their distingushed talks:

- P. Kars - Utopics BV, "Formal Methods in the Design of a Storm Surge Barrier Control System"
- U. Herzog - Univ. of Erlangen-Nurnberg, "Stochastic Process Algebras for Qualitative Assurance"
- R. Mazzeo - SASIB Railway S.p.A., "Vital Processor Interlocking: A Case Study of Utilisation of Form Methods for all the Design Phases"
- G. Mongardi ANSALDO-Trasporti, "Formal Methods for Raylway Signalling Applications: rationale an case studies"

The next ERCIM/FMICS workshop will be held in Amsterdam on May 25-26 1998 (look at the pa `http://www.cwi.nl/~luttik/FMICS/index.html`).

For more information on the series of workshops ERCIM/FMICS please look at the page `http://fdt.cnuce.cnr.it/~latella/FMICS/WgDescription.html`

D. Latella
CNR/CNUCE
FMICS-WG Chair

S. Gnesi
CNR/IEI
FMICS 97 PC Chair