REPORTS ON CONFERENCES

Report of the 11th British Colloquium for Theoretical Computer Science

University College Swansea, Wales, 2-5 April 1995
(Sponsored by the Engineering and Physical Sciences Research Council and by Praxis)

The meeting was held in Clyne Castle set in an award winning park with views over Swansea Bay, the Mumbles and the Gower Peninsula, an area of outstanding natural beauty. This report covers the technical content of a very successful meeting. An independent review of BCTCS11 has already appeared in the EATCS Bulletin (Number 56, June 1995). The following pages list the titles of talks by distinguished guests and the titles and abstracts of contributed talks.

BCTCS12 will be held in the University of Kent, 1-4 April 1996 under the local Chairmanship of Simon Thompson. Details are on www page http:\\www.ukc.ac.uk\computer_science\Bctcs12\ or available by e-mail from S.J.Thompson@ukc.ac.uk or J.Derrick@ukc.ac.uk. We warmly encourage contributed talks from national and overseas participants at this lively but relaxed foremost British Theoretical Computer Science conference. We welcome unrefereed reports of on-going research and from research students who will particularly benefit from the series of tutorial seminars which are on integral part of the meeting.

Alan Gibbons and Chris Tofts

BCTCS National Committee: Alan Gibbons (Chairman, Warwick), Paul Dunne (Secretary, Liverpool), Iain Stewart (Treasurer, Swansea), Julian Bradfield (Edinburgh), Savita Chauhan (Doctoral student member, Swansea), Mike Holcombe (Sheffield), John Stell (Keele), Simon Thompson (Kent), Chris Tofts (Manchester), John Tucker (Swansea).

Invited Talks

Metric Semantics
J.W. de Bakker, e-address: jaco@cwi.nl
CWI/VUA, Amsterdam

Algebraic specifications and proofs by induction
Jan Heering, e-address: Jan.Heering@cwi.nl
CWI/VUA, Amsterdam

Higher-Order Processes and Their Models
M. Hennessy, e-address: matthewh@cogs.susx.ac.uk
Univerity of Sussex

Counting the Inhabitants of a Type
Roger Hindley, e-address: J.R.Hindley@swansea.ac.uk
Mathematics Department, University of Wales, Swansea, Swansea SA2 8PP, U.K.

The Computational Complexity of Bisimilarity
Faron Moller, e-address: fm@sics.se
Swedish Institute for Computer Science

Categories, Proofs and Games.
D.E. Rydeheard, e-address: david@cs.man.ac.uk
Department of Computer Science, the University, Manchester, M13 9PL

Descriptive Complexity Theory
Iain Stewart, e-address: I.A.Stewart@swansea.ac.uk
Department of Computer Science, University College Swansea, Swansea, SA2 8PP

Contributed Talks

$*$-realizations of deterministic finite automata

Andries van der Walt and Lynette van-Zijl, e-addess: lynette@cs.sun.ac.za
Stellenbosch University, 7600 Stellenbosch, South Africa

Since the introduction in 1959 of nondeterminism in finite automata by Rabin and Scott [RS], quite a number of different devices for the achievement of 'succinctness' or 'power' have been defined. A (partial) list includes various two-way capabilities, alternation, bounded cooperative concurrency and combinations of these. One purpose of this paper is to give a unifying framework for (most of) them. Flowing from this, many other devices of this nature are shown to exist, and some of these are explored briefly.

Priority-Queue Techniques for Visibility Computations

Frank Devai, e-address: fl.devai@ulst.ac.uk
School of Computing and Mathematics, University of Ulster, Londonderry, UK, BT48 7JL

A worst-case optimal algorithm is proposed for determining the visibility of N line segments in the plane. The method is generalised to determine the visibility of a collection of pairwise disjoint polygons with a total of N edges in three-dimensional space. The three-dimensional algorithm takes $O((N+k)logN)$ time and $O(N+k)$ space, where $k, 0 <= k <= N(N-1)/2$, is the total number of intersecting pairs among the images of the edges in the projection of the input polygons onto a plane perpendicular to the viewing direction.

Can Implicit Methods Deliver Efficient Programs?

H. James Hoover. e-address: hoover@cs.ualberta.ca
University of Alberta, Edmonton, Alberta, Canada T6G 2H1

This talk is a progress report on the Mizar-C project to evaluate the feasibility of dealing with resource consumption in implicit programming methods. The most important feature of Mizar-C is its accounting of the resources required to compute. Statements whose validity has been established by non-constructive reasoning, such as excluded middle, are marked as non-computable, and so have no computational content. Constructively established statements are marked according to the resources consumed when executing their computational content. Thus Mizar-C permits us to employ classical reasoning, except when we want to perform computations and extract programs. Mizar-C's second significant feature is that all typing in done via predicates. This lets us cast the objects of our reasoning into different types.

Parallel algorithm for the conversion of a regular expression to its Glushkov's automaton.

Djelloul Ziadi and Jean-Marc Champarnaud, e-address: ziadi@dir.univ-rouen.fr
Universite de Rouen, Place Emile Blondel, 76821 Mont-Saint-Aignan, Cedex

Rytter proved that the transformation of a regular expression of size s to a corresponding non-deterministic finite automaton with ϵ-transitions can be achieved in $O(logs)$ time using $O(s/logs)$ processors on a P-RAM. We prove that the transformation of a regular expression E of size s to its automaton of Glushkov which is a non deterministic automaton without ϵ-transitions can be done in $O(logs)$ time on a P-RAM under the condition that E is in star normal form. The generated automaton has $n+1$ states, where n is the number of the occurences of letters in the expression E. On the other hand we show that the star normal form of regular expression can be calculated from E in $O(logs)$ time with $O(s)$ processors on a P-RAM.

Subclasses can be Subtypes

Sophia Drossopoulou & Dan Yang, e-address: scd@doc.ic.ac.uk
Department of Computing, Imperial College, London SW7

We develop a type system for object oriented programming languages, in which all subclasses are considered as subtypes without any restrictions on the types of arguments of the methods redeclared in the subclasses. Also, functional dependencies between the type of the receiver and/or argument and the type of the result can be expressed. This paper addresses the soundness of this type system.

Verification and Validation of Attribute Grammars

Bernhard Bauer, e-address: bauer@informatik.tu-muenchen.de
Institut fuer Informatik, Technische Universitaet Muenchen, 80290 Muenchen

Attribute grammars as introduced by Knuth are a well accepted tool, e.g. for specifying compilers, language-based environments, user interfaces, document architecture and (static) semantics of programming languages. But there are only a few papers appeared on the topic of verification and validation of attribute grammars. By specifying the semantic functions algebraically, i.e. by axioms, theorem proving techniques can be applied in the framework of attribute grammars, since this unifying formalism allows the definition of a model theoretic semantics for attribute grammars. **Attributed term induction**, developed as a new proof principle for attribute grammars, can be used to prove properties between attribute instances at distinguished nodes of a special non-terminal in the syntax tree. The new approach treats inherited and synthesized attributes in a common fashion and can be seen as a general proof principle for attribute grammars.

Correctness Issues in Transformational Refinement

Peter Kilpatrick, M Clint, T J Harmer and S Fitzpatrick, e-address: peter@plk.cs.qub.ac.uk
Dept. of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, UK

Transformational refinement in the context of this talk refers to the *automatic* transformation of a specification expressed in a functional programming language (typically SML, LISP or Miranda) to an efficient implementation expressed in an imperative programming language (typically FORTRAN or one of its parallel derivitives). In particular, our work focuses on the derivation of efficient numerical mathematical algorithms for execution on a range of parallel machines.
A transformational derivation proceeds not as a monolithic translation process (c.f. compilation) but rather as a sequence of sub-derivations which convert a program via a number of intermediate forms which lie between its specification expressed in a specification language and its implementation expressed in the target language. Each of these transitions may be sub-divided into further intermediate forms. In this talk we consider issues relating to the correctness of this transformation system. We first establish what we mean by correctness in this context and then describe a framework in which correctness proofs may be undertaken.

Case Studies in Higher-Order Algebra: The Specification and Verification of a Dataflow Algorithm.

L.J.Steggles, e-address: L.J.Steggles@newcastle.ac.uk
Computer Science Department, University of Newcastle, Newcastle upon Tyne, NE1 7RU.

Higher-order algebra provides a natural framework in which to formally develop computing systems and has been shown to be substantially more expressive than first-order algebraic methods. We present a case study of higher-order algebraic methods applied to the specification and verification of a dataflow algorithm for computing the Hamming stream. This case study demonstrates the expressive power of higher-order algebraic methods by making use of a non-constructive task specification which specifies the characteristic properties of the Hamming stream rather than how to compute it. In order to do this we use a discontinuous (with respect to the Tynchonoff topology) search function which is non-constructive and it is here that the additional expressive power of higher-order algebraic specification methods is required.

Local Confluence for Strong Reductions with Categorical Combinators

Nicholas Merriam, e-address: nam@doc.ic.ac.uk
Imperial College, University of London.

Curien presented the language of categorical combinators in his book, Categorical Combinators, Sequential Algorithms and Functional Programming. The decidability of weak equality for terms in typed categorical combinatory logic is simple. However the strong laws for deciding extensional equality form a rewrite relation which is not locally confluent. Standard techniques such as the Knuth Bendix procedure for completion of rewrite systems, and class rewriting are unable to provide a solution. Happily, critical pair analysis of the rewrite rules suggests a fairly simple remedy: one rule, sL, must be applied not just with the pattern matching rewrite rule of inference but with S-matching, where S is a strict subset of the whole rule set. By S-matching we mean matching modulo a set of equations S. We present an algorithm for S-matching this particular rule and use it to build a decision procedure for extensional equality of typable terms in categorical combinatory logic.

Infiniteness of proof(α) is Polynomial-Space Complete

Sachio Hirokawa, e-address: s.hirokawa@swansea.ac.uk
Department of Mathematics, University of Wales, Swansea SA2 8PP

Infiniteness problem is investigated for the set *proof*(α) of closed λ-terms in β-normal form which has α as their types. According to 'Formulas-as-types' correspondence, it is identical to the set of normal form proofs for α in intuitionistic logic. Firstly we show that the infiniteness is determined by checking λ-terms with the depth at most $2|\alpha|^2$. Thus the problem is solved in polynomial-space. Secondly we prove the polynomial-space completeness of the problem by reducing the emptiness problem, which is know to be polynomial-space complete, to the infiniteness problem.

Symbolic Verification of Hardware Systems

Howard Barringer, Graham Gough, Brian Monahan & Alan Williams
e-address: alanw@cs.man.ac.uk
Department of Computer Science, University of Manchester, Manchester, UK

One approach commonly used for establishing the behavioural equivalence of hardware systems uses state-space exploration to establish a *bisimulation* relation between the systems, modelled as *labelled transition systems.* It has the distinct advantage of being automatic, and can produce counter-example information when a verification fails. A second method represents system behaviour using logical expressions, and then establishes equivalence by employing theorem-proving techniques. It operates abstractly at a high level, thus not necessarily suffering from combinatorial explosion. Further, when two systems are shown to be different, it may be possible to give structured debugging feedback information at the level of the initial system representation. We introduce a novel verification approach which effectively merges the above techniques. We take Park and Milner's standard (strong) bisimulation equivalence as our notion of equivalence between labelled transition systems. However, we present the transition systems abstractly as *deterministic machines* and then define a *state bisimulation* relation between the state spaces of the two machines. The analysis proceeds in a truly symbolic manner, at the level at which the design is expressed. The approach described offers the possibility of containing the usual growth in complexity of verification, whilst maintaining a high degree of automation.

A Note on Confluence of Term Rewriting Systems under Joinability of Critical Pairs in One Step of Parallel Reduction

Mauricio Ayala, e-address: ayala@alpha.mat.unb.br
Departamento de Matemática, Universidade de Brasília

Termination and confluence are the key properties of term rewriting systems (TRSs). Many criteria for proving termination as well as confluence of terminating TRSs have been developed. Without termination assumption, criteria to guarantee confluence are either very restrictive or lost their practical interest.
We study the following open question, *Are confluent* LL *TRSs for which for every CP* $< P, Q >$, $Q \nrightarrow\!\!\!\!\!\parallel P$*?* Our investigation, on this very simple stated open question, contributes in giving some light to understand its real complexity, but is not enough to conjecture about either a positive or negative answer.

Directed Acyclic Graphs as Data Types for Concurrent Object-Oriented Programming

Dorel Lucanu, e-address: lucanu@uaic.ro
University "Al. I. Cuza", Department of Computer Science, Berthelot 16, 6600 -Iaşi, Romania.

Graph transformation have been studied and applied in many fields of computer science: formal languages theory, software engineering, concurrent and distributed systems, etc. Three tools have been proposed for studying graph transformations: mathematical logic, universal algebra, and category theory. Here we briefly present the three approaches: A logical formula describe graph properties and so it defines the set of graphs which satisfy these properties. The category theory is used for specifying graph rewriting rules and for defining the initial solution of a system of graph equations. Universal algebra allows to build larger graphs from smaller ones by means of operations and to represent graphs by finite algebraic expressions. We offer a mathematical tool for treating in an uniform way directed acyclic graphs (on short dags). Our approach unifies the last two above tools: category theory and universal algebra. We define several $2W$-algebras of (un)labelled directed acyclic graphs (on short dags) and prove that our approach is enough powerful to manipulate complex dags.

Infiniteness of proof(α) is Polynomial-Space Complete

Sachio Hirokawa, e-address: s.hirokawa@swansea.ac.uk
Department of Mathematics, University of Wales, Swansea SA2 8PP

Infiniteness problem is investigated for the set $proof(\alpha)$ of closed λ-terms in β-normal form which has α as their types. According to 'Formulas-as-types' correspondence, it is identical to the set of normal form proofs for α in intuitionistic logic. Firstly we show that the infiniteness is determined by checking λ-terms with the depth at most $2|\alpha|^2$. Thus the problem is solved in polynomial-space. Secondly we prove the polynomial-space completeness of the problem by reducing the emptiness problem, which is know to be polynomial-space complete, to the infiniteness problem.

An $\Omega(\log n)$ time linear cost lower bound for the single function coarsest partition problem

Clive Galley, e-address: clive@dcs.kcl.ac.uk
Department of Computer Science, Kings College London

We consider the single function coarsest partition problem for a set S, where $|S| = n$, and a function from S to S. We present an $\Omega(\log n)$ time, linear work, lower bound for this problem on the common CRCW PRAM model of computation.

Genetic Program Systems as Generic Problem Solvers

Brian Stonebridge, e-address; brian@compsci.bristol.ac.uk
Department of Computer Science, University of Bristol, Bristol

It is remarkable that in just twenty years, the Genetic Algorithm (GA) community has proceeded from mutation as the sole form of change to contemplate automatic ways to solve problems hierarchically by decomposition into subproblems, by solving subproblems by GAs and assembling the results. The ideas of automatic function definition and the emergent genetic programming are essentially products of work during the 80s published in the 90s. We are now in a position to search for highly fit computer programs in the space of all possible computer programs. We present some basic theory and the results of experiments in this field.

Language Spaces

Chris Holt, e-address: chris.holt@ncl.ac.uk
Department of Computer Science, University of Newcastle

We are used to characterizing semantics domains as mathematical objects, spaces containing points with various relations among them; and we are used to viewing syntactic domains as structures in spaces with given relations. For instance, textual language spaces are linear sequences, with points associated with characters from a finite alphabet. These are mapped into spaces consisting of linear sequences of symbols; and these are mapped into spaces of (parse) trees of semantic objects. Graph grammars require syntactic spaces to be graphs; Petri nets require their syntactic spaces to be bi-partite graphs; and so on. It is suggested that a language framework should allow a user to specify the characteristics of a language space, before embedding a statement in that space. The interfaces among different kinds of spaces should then be defined to facilitate the connection of objects written in different kinds of languages. The goal is to allow 1D, 2D, and 3D statements to be linked together in program structures.

Symbolic Verification of Hardware Systems

Howard Barringer, Graham Gough, Brian Monahan & Alan Williams
e-address: alanw@cs.man.ac.uk
Department of Computer Science, University of Manchester, Manchester, UK

One approach commonly used for establishing the behavioural equivalence of hardware systems uses state-space exploration to establish a *bisimulation* relation between the systems, modelled as *labelled transition systems*. It has the distinct advantage of being automatic, and can produce counter-example information when a verification fails. A second method represents system behaviour using logical expressions, and then establishes equivalence by employing theorem-proving techniques. It operates abstractly at a high level, thus not necessarily suffering from combinatorial explosion. Further, when two systems are shown to be different, it may be possible to give structured debugging feedback information at the level of the initial system representation. We introduce a novel verification approach which effectively merges the above techniques. We take

Park and Milner's standard (strong) bisimulation equivalence as our notion of equivalence between labelled transition systems. However, we present the transition systems abstractly as *deterministic machines* and then define a *state bisimulation* relation between the state spaces of the two machines. The analysis proceeds in a truly symbolic manner, at the level at which the design is expressed. The approach described offers the possibility of containing the usual growth in complexity of verification, whilst maintaining a high degree of automation.

Completeness of a Simple Program Transformation Framework

Peter Burton, e-address: Peter.Burton@dcs.qmw.ac.uk
Computer Science Department, Queen Mary and Westfield College, London University.

A transformation framework is *complete* if, for any equivalent programs P and Q, it allows a derivation of P from Q. In this context, completeness questions are seldom addressed. The reason is plain: for any universal programming formalism, equivalence is not semi-decidable; but a complete transformation framework yields a semi-decision algorithm by the enumeration of all derivations; hence completeness can immediately be ruled out. This still leaves the possibility of positive results, provided we are willing to consider formalisms which cannot express all computable functions. We consider here, in essence, one-level primitive recursion – but slightly generalised. We have a category whose objects are derivors, i.e. programs in this extremely limited sense. Because these arrows arise only between equivalent programs, we look only at sub-categories in which all objects are equivalent. In particular, let us consider the full sub-category comprising *all* objects equivalent to some given F. Various questions can be asked about this: for example, does it possess finite products? A positive answer to this question for all F would be sufficient (though not necessary) for the completeness of our transformation framework. We shall show that a positive answer indeed results, and indicate some (cautious) extensions of the framework beyond this first version.

Optimal Prefix String Matching and Covering in Two Dimensions

Maxime Crochemore, Costas S. Iliopoulos & Maureen Korda, e-address: csi@dcs.kcl.ac.uk
Department of Computer Science, King's College London, Strand, London.

The problem of prefix string matching in two dimensions is to compute the largest prefix of a square matrix P that occurs at each position of another given matrix T; a linear algorithm is shown for this problem, improving previous results by Giancarlo and Grossi by a factor of $O(\log|T|)$. The new algorithm makes use of a two-dimensional "failure function", whose computation is also shown to require linear time. A *cover* is a generalisation of the notion of the period of a given string x. A substring w is a *cover* of x if x can be constructed by concatenations and superpositions of w. We consider the problem of computing all the square *covers* of a given two-dimensional square pattern; a linear algorithm for this problem will be presented.

Tests that find all faults (well almost!).

M.Holcombe & Florentin Ipate, e-address: M.Holcombe@dcs.shef.ac.uk
Department of Computer Science, Sheffield University, Sheffield.

We present a method of generating test sets based on formal specifications that detect all faults subject to no faults existing in specific components (basic processing functions). The work is based on a theory of testing formulated for general computable functions and has been applied to a number of practical case studies. Current testing methods can say nothing about the possibility of faults remaining in a system (software) on successful completion of the test procedure. This method is the first approach that overcomes this objection to testing. it reprsents and new type of reductive testing method. Specific testability conditions are defined which also provide a theoretical framework for the problem of constructing testable systems.

Reflection and Control in Production Systems

Iain D. Craig, e-address: Iain.Craig@dcs.warwick.ac.uk
Department of Computer Science, University of Warwick, Coventry CV4 7AL

Production systems are common in AI. They frequently employ upon general-purpose, domain-independent, control techniques. These techniques are essentially syntactic: they can be insensitive to the demands of particular problem types. We have specified (in Z) and constructed a production rule interpreter called ELEKTRA which can also support an arbitrary number of meta levels. It

allows access to its interpreter's functions and state, and thereby permits the definition of rule interpreters structured as sets of rules. At runtime, ELEKTRA permits multiple rule interpreters to be present: the interpreters compete for processor time on a uni-processor implementation, so only one will be active on any cycle. The choice of which interpreter to activate can be controlled by meta rules.ELEKTRA allows its own interpreter to be encoded as a rule set and executed. In this paper, we will describe ELEKTRA and some of the theoretical problems it raises, and show how different rule interpreters can be encoded. We will also discuss some of the ways in which parallel computation can be performed.

Program improvement by proof planning

Peter Madden, e-address: madden@mpi-sb.mpg.de
Max-Planck-Institut fuer Informatik, Im Stadtwald, W-66123 Saarbruecken, Germany.

A uniform framework for synthesizing efficient programs, based on inductive theorem proving, is under development. In this framework we hope to capture a diverse range of program transformation techniques and apply them to the problem of improving the efficiency of programs. A prototype system has been implemented, and some transformation techniques have been added. (The work described in this article was carried out in collaboration with Alan Bundy, Ian Green and Jane Hesketh, all of the Department of AI, University of Edinburgh, Scotland.)

Efficient implementation of constructive type theory

Simon Thompson, e-address: S.J.Thompson@ukc.ac.uk
Computing Laboratory, University of Kent

Constructive type theory is simultaneously a functional programming language and a constructive logic. This talk introduces a version of Martin-Lof type theory as a programming tool and, after examining some simple example programs and proofs, discusses the problem of 'computational relevance' of the proof information contained in definitions. We then show how to use ideas of abstract interpretation to detect cases of redundant proof information, and so give a sound basis for program transformations to remove these parts which contribute to program execution being less efficient than it might be.

Documenting Systems using Tables and Equations.

Anthony Wilder, e-address: csanton.pg@swansea.ac.uk
Computer Science Department, University College Swansea, Swansea, SA2 8PP, Wales

A new Tabular method for documenting systems is presented. This method is based on concepts developed by D.L. Parnas and others at McMaster University, Canada. By considering systems modelled by event triggered state transition functions a tabular description can be systematically constructed from the observed behaviour of the system. These notions will be illustrated by various case studies.

The case studies gravitate towards User Interfaces modelled by event triggered state transition functions. We shall see a variety of User Interface mechanisms tabular documented.

The mathematical relationship between tables, equations and algebras will also be addressed. We shall describe the syntax ("form") of two types of Parnas tables and define a semantics ("behaviour") using algebras.

Validation of Reactive Programs which Incorporate Structured Data

H.L. Loedolff, P.J.A. de Villiers & W.C. Visser , e-address: visserw@cs.man.ac.uk
Department of Computer Science, The University, Manchester, M13 9PL

It is a challenging task to design a reactive program that can be proven to be correct. Model checking provides a promising technique for building validation models of reactive programs. Such models can be shown to have important correctness properties. Several validation languages have been developed for the purpose of validating protocol models. These validation languages model data structures in an abstract way in order to prevent a state explosion. However, to validate models of microkernels more detailed structured data types are essential. In this paper we describe a compaction technique that allows efficient model checking of structured data. An experimental validation language ESML is described and used to develop a validation model of a process scheduler which illustrates the use of structured data types. The process scheduler model was derived from a larger model of a real microkernel.

Synthesis of Box Expressions from Petri Boxes

Martin Hesketh, e-address: Martin.Hesketh@newcastle.ac.uk
Computing Science Department, University of Newcastle, Newcastle upon Tyne, UK

The Petri Box Calculus is a Petri net based model for specifying concurrent systems. The calculus consists of an algebra of Box Expressions, and a corresponding algebra of Petri Boxes (labelled Petri nets). The algebra of Petri Boxes provide a compositional semantics for Box Expressions, and a means of translating an expression into a labelled Petri net. The reverse process - the problem of synthesising a Box Expression from a Petri Box, in addition to having practical applications, can be used as a tool to show an axiomatisation of the Box Calculus is complete.

This talk describes an investigation into the synthesis problem, and presents a solution based on a collection of synthesis rules. Each rule has a set of preconditions which must be satisfied for the rule to be applied. The preconditions are based on structural properties of the net. The application of a rule refines the Box Expression being synthesised, and decomposes the net into a collection of subnets. The synthesis process is applied recursively to each of the subnets, to yield the synthesised expression.

A Comparison of the Partial Matching Performance between Correlation Matrix Memory and Multi-Level Superimposed Coding.

Richard Filer and James Austin, e-address: rjhf@minster.york.ac.uk
Department of Computer Science, University of York, York YO1 5DD.

We presents the results of an analysis of the partial matching characteristics of Correlation Matrix Memory. CMM is a binary, neural network, associative memory and is a development of the ADAM system. We are particularly interested in the application of CMM to the fast partial matching of tokens representing rules, i.e., using CMM as the inference engine of an expert system. It is important to quantify the partial matching performance of CMM and useful to make a comparison with a conventional partial matching method, Multi-level Superimposed Coding. CMM partial matching performance can be shown to depend principally on the number of true matches in the database, which is a desirable property. We explain in detail the analysis and consider possible time savings in other aspects of CMM function (for example, CMM may be implemented in hardware).

Semantics of Serializable Database System by using Formal Models

Haruo Yamaguchi, e-address: yamaguch@cs.man.ac.uk
Manchester University

This is an experiment to capture the semantics of concurrent serializable database system by using formal methods. The difficulty resides in the description of interference among transactions and consistency of database as a global resource of the system. Traditional serializable database theories are based on graph-theoretical approaches, and in this article, some formal approaches are employed including Structural Operational Semantics to give semantics to the concurrent serializable database system without refering graph-theoretical approaches. Application for distributed database systems are also discussed.

Empiricism in Computer-based Modelling

Meurig Beynon— & Paul Ness, e-address: pen@dcs.warwick.ac.uk
Department of Computer Science, University of Warwick, Coventry CV4 7AL

In connection with our Empirical Modelling Project, we have developed case-studies and software tools for constructing models of real-world phenomena. These exploit agent-oriented modelling over spreadsheet-like representation of state and lead to environments for simulation that allow rich modes of interaction only constrained (in the same way that interaction with a spreadsheet is constrained) by the interpretation of variables and their interrelationships. Such freedom for action has many advantages and in particular permits scientific experimantation.

In our view, the modelling methods we have developed are of interest in connection with the foundations of computer science because they give insight into how computers can support the empirical activity that in particular informs theory development. This has implications for many areas of computing.

By directly associating variables with real-world observables, our modelling establishes an unusually close relation between a model and its real world semantics, but makes its operational semantics enigmatic. Understanding our models in relation to theoretical models currently being developed for concurrent system specification is a challenging problem.

Program schemes and polynomial time

Savita Chauhan, e-address: s.r.chauhan@swansea.ac.uk
Department Computer Science, University of Wales Swansea, Swansea SA2 8PP

We define classes of program schemes with high-level programming constructs, such as stacks, and which take finite structures as inputs, and show how these classes of program schemes characterize complexity classes. These high-level characterizations enable us to vary whether constructs, such as a successor relation or a linear order, are included in our program schemes, and to hopefully prove inexpressibility results for certain problems and to investigate refined classes of such program schemes. This research is still in a preliminary stage.

Algebraic Operational Semantics

Karen Stephenson, e-address: cskarens.pg@swansea.ac.uk
Department of Computer Science, University of Wales, Swansea SA2 8PP, UK.

We present an algebraic approach to define the operational semantics of programming languages. We show how the behaviour of a system can be described using a function $Comp(P, \sigma, t)$ which gives the state of the computation of a program P on an initial state σ at time t. Given a set of atomic instructions over which all other programs are constructed, the *Comp* function is axiomatically defined for any system. This approach has the advantage over many other semantic models by being modular, computable, axiomatic and not being higher-order.

We outline how this methodology can be used in proving general propositions of programs and in the correctness of compilation.

RNC Algorithms for the Uniform Generation of Paths and Trees in Graphs

Ida Pu, Michele Zito, Martyn Amos and Alan Gibbons, e-address: amg@dcs.warwick.ac.uk
Department of Computer Science, University of Warwick

We outline a number of randomised parallel algorithms for the uniform generation of variously defined paths and for the uniform generation of spanning trees of a connected simple graph. The algorithms run in low-order polylogarithmic time and involve a polynomial amount of work. These are essentially the first efficient parallel algorithms described for these problems.

Uniform Parallel Generation of Combinatorial Structures

Michele Zito, Ida Pu, and Alan Gibbons, e-address; amg@dcs.warwick.ac.uk
Department of Computer Science, University of Warwick

We describe an RNC algorithm for the uniform generation of unlabelled undirected graphs with a specified number of nodes. The algorithm is archetypal in the sense that many other RNC algorithms for the uniform generation of other combinatorial structures are possible using the same framework. The framework is based upon a parallisation of a result of Dixon and Wilf and uses simple probabilistic analysis to convert the expected running time of a sequential algorithm into a worst case parallel running time.

A Fast Approximation Algorithm for the Multicommodity Flow Problem

Tomasz Radzik, e-address: radzik@dcs.kcl.ac.uk
Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK

In this paper we consider the following optimization version of the multicommodity flow problem: Find the minimum possible number OPT and a flow of all commodities which is feasible when all edge capacities are multiplied by OPT. Such a flow is called an optimal (concurrent) flow. A flow is t-optimal if it is feasible when all capacities are multiplied by $(1+t)OPT$. The fastest algorithm for computing an optimal flow (Vaidya) runs in $O(k^{3.5}n^3m^{0.5}\log(nB))$ time, n is the number of vertices, m the number of edges, k the number of commodities and B the largest demand/capacity (the capacities and demands are integral). A t-optimal flow, for a given constant t, can be computed considerably faster. The randomized version of the algorithm of Leighton, Makedon, Plotkin, Tardos, and Tragoudas finds a t-optimal flow after $O(k \log k \log n)$ minimum-cost flow computations, while the deterministic version requires $O(k^2 \log k \log n)$ such computations. One minimum-cost flow computation takes $O(nm \log^2 n)$ time.

We show a deterministic algorithm which computes a t-optimal flow using only $O(k \log k \log n)$ minimum-cost flow computations. The total runing time is therefore $O(knm \log k \log^3 n)$. Our algorithm follows the general scheme of Leighton's et.al. randomized algorithm but the random selection of the commodities is replaced by the deterministic round-robin.

Constructivity in Classical Logic

Charles Stewart, e-address: Charles.Stewart@comlab.ox.ac.uk
Programming Research Group, Oxford University.

The Curry-Howard correspondence fundamentally links logic and computation, but is restricted to intuitionistic logics. Recently progress has been made that demonstrates that we can also expect such a relationship for classic logics. Girards linear logic is a wholly constructive logic in which (cut-free) classical logic can be satisfactorily embedded. More concretely Griffin in 1990 published a successful paper outlining a method of viewing continuations (as in Scheme, ML) as effectively modeling the absurdity rule.

Despite these successes, they leave unanswered many fundamental questions about the nature of constructivity in classical logic. Parigots system of Free Deduction has many desirable properties from a proof theoretic perspective; most importantly it was the first calculus of classical logic with a confluent and strongly normalising cut-elimination procedure, required for the proof theory to give a notion of computational value. Finally I present a correspondence between proofs in Free Deduction and a subset of the constant-enriched λ-calculus, and instantiations of the constants within the pure untyped λ-calculus.

Strategies for Temporal Resolution

Clare Dixon, e-address: dixonc@cs.man.ac.uk
Department of Computer Sceince, The University of Manchester

An approach to the mechanisation of temporal logics, based on a form of clausal resolution has been developed by Fisher. Temporal formulae incorporating both past-time and future-time operators are converted to a normal form, then both step and temporal resolution rules are applied. The temporal resolution rule attempts to match eventualities which must be satisfied with sets of rules which together imply that the eventuality will never be satisfied.In the talk we shall describe Fisher's resolution process, mention the different algorithms developed to enable the application of the temporal resolution rule, and outline and discuss the strategies proposed for each part of the temporal resolution process.

INDUCT: A Logical Framework for Induction Over Natural Numbers and Lists Built in SEQUEL

Andrew A Adams, e-address: aaa@dcs.st-andrews.ac.uk
School of Mathematical and Computational Sciences, North Haugh, St Andrews, KY16

SEQUEL is a new language built on top of Lisp and designed for implementing theorem provers and proof assistants. It includes a type theory for Lisp (XTT — eXtended Type Theory) and functions for sequent calculus and rewriting systems. The project described involves a description of a framework for proving properties of functions, and the implementation details of programming such a system in SEQUEL's sequent calculus notation. The framework, called INDUCT, is based on the theory laid out in Boyer and Moore's 1979 Book 'A Computational Logic'. INDUCT has been limited to producing a framework dealing with Natural Numbers and Lists. Extension of the framework to other Lisp types and then to user-defined SEQUEL types should be possible, since little of the framework is dependent upon the nature of the types, merely upon the theoretical requirements common to INDUCT and NQTHM. Areas which would require more than rudimentary changes to achieve these ends will be highlighted and suggestions made as to how these changes could be implemented.

Order Sorted Theories

John Stell, e-address: John@cs.keele.ac.uk
Department Computer Science, Keele University, Keele, Staffs, ST5 5BG

Order sorted algebra is the generalization of many sorted algebra obtained by having a partially ordered set of sorts rather than merely a set. It has been applied to the handling of errors in abstract data type specifications. There are several variants of order sorted algebra, and relationships between these are known. However, the question of what should be meant by an order sorted theory, as opposed to a presentation given by a particular signature of operation symbols, does not seem to have received much attention. In this talk I will show how the notion of a theory as a category, in the sense of Lawvere, fits order sorted algebra by having poset enriched theories, and taking as models suitable functors to categories of sets and partial functions.

Two Topologies Are Better Than One

Simon John O'Neill, e-address: S.J.ONeill@dcs.warwick.ac.uk
Department of Computer Science, University of Warwick

We consider an appropriate context in which to consider these spaces is as a bitopological space, i.e. a space with two (related) topologies. From this point of view, we cover the groundwork for a theory of partial metric spaces by generalising ideas from topology and metric spaces.

For intuition we repeatedly refer to the real line with the usual ordering and metric. However we also consider other examples of more relevance to Computer Science.

Timed Synchronous Interaction Categories

Justin Pearson, e-address: justin@dcs.rhbnc.ac.uk
Department of Computer Science, RHBNC, London.

This talk presents a Timed Extension of Abramsky's Interaction Categories. An Interaction Category is a new type theorectic way of looking at process algebras, where instead of having processes as objects and morphisms as structure preserving maps between processes, the objects of Interaction Categorys are saftey specifications and the morphims are processes satisfying certain saftey specifications. Timed Interaction Categories, include timing information in the types, in the synchronous version a timed synchronisation algebra captures how timed processes act under parallel composition.