Report of the 9th British Colloquium on Theoretical Computer Science

University of York, 28-31 March 1993

The venue for BCTCS9 for 1993 was provided by the University of York which has one of the most attractive university campuses in the United Kingdom located alongside one of its most delightful cities. Breaks in conference proceeds found many participants exploring the famous Cathedral Minster and investigating the Roman and Viking past of this historic City of York. The delights of the York campus, set about by parkland, lakes and wildlife, made a particularly relaxed venue for the meeting. The conference dinner provided the opportunity for participants to thank York's hard-working organising committee of Hussein Zedan (Chair), Alan Dix, Chris Ho-Stuart and Ming Fang.

The Colloquium was admirably supported this year by nearly forty papers of wide diversity. In particular, the invited talks of five distinguished guests (Michael Atkinson, John Lloyd, Gordon Plotkin, Jean-Marc Steyaert and Bob Tennant) added particular interest and strength to the programme. Abstracts of all the talks may be found below.

In 1994, BCTCS will be held in March at the University of Bristol, UK. Bristol's organising Chairman is Brian Stonebridge who should be contacted for preliminary enquiries (e-mail address: Brian.R.Stonebridge@uk.ac.bristol). The BCTCS committee plan that this will be a special meeting to mark the tenth anniversary of the Colloquium. The following year, 1995, the meeting will take place at the University of Swansea, Wales.

Alan Gibbons,
Chairman BCTCS

Abstracts of Invited Talks

The Capability of Priority Queues as Data Transformers

Michael Atkinson (St Andrews University)
email: mda@cs.st-and.ac.uk

Abstract data types which support an Insert and Delete operation (with possible restrictions) transform input sequences (which are inserted in sequential order) into output sequences (obtained by the results of successive deletions). In the case of the abstract data type 'Queue' the relationship between the input sequence and output sequence is trivial while for a 'Stack' it is well understood. This lecture surveys recent work on this relationship in the case of a 'Priority Queue', gives a number of combinatorial results, and indicates the connections with various types of tree.

Declarative Logic Programming

John W. Lloyd (University of Bristol)
email: jwl@compsci.bristol.ac.uk

We begin with a brief assessment of the current state of logic programming (LP). LP has had a substantial influence of many parts of Computer Science, including databases, artificial intelligence, and natural language processing. It also has a well-developed theory and significant potential for the exploitation of parallelism. On the other hand, LP has had little commercial impact and the main LP language, Prolog, has many deficiencies. I will identify several causes for these problems, concentrating particularly on the curious reluctance of many researchers to take the declarative aspects of LP sufficiently seriously.

We then briefly review the programming language Gödel, which is specifically intended to address some of the problems mentioned above. Gödel is a declarative, general-purpose programming language in the family of logic programming languages. It is a strongly typed language, the type system being based on many-sorted logic with parametric polymorphism. It has a module

system. Gödel supports infinite precision integers, infinite precision rationals, and also floating-point numbers. It can solve constraints over finite domains of integers and also linear rational constraints. It supports processing of finite sets. It also has a flexible computation rule and a pruning operator which generalises the commit of the concurrent logic programming languages. Considerable emphasis is placed on Gödel's meta-logical facilities which provide significant support for meta-programs that do analysis, transformation, compilation, verification, debugging, and so on. The declarative nature of Gödel makes it particularly suitable for use as a teaching language; narrows the gap which currently exists between theory and practice in logic programming; makes possible advanced software engineering tools such as declarative debuggers and compiler generators; reduces the effort involved in providing a parallel implementation of the language; and offers substantial scope for parallelization in such implementations.

We conclude with some remarks about future directions for logic programming language design.

A Logic for Parametric Polymorphism

Gordon Plotkin (University of Edinburgh)
email: gdp@dcs.ed.ac.uk

We introduce a logic for parametric polymorphism. Just as LCF is a logic for the simply-typed λ-calculus with recursion and arithmetic, our logic is a logic for Girard's System F. The logic permits the formal presentation and use of Reynold's relational parametricity. Parametricity yields—for example—encodings of initial algebras, final co-algebras and abstract datatypes, with corresponding proof principles of induction, co-induction and simulation.

On the Average Complexity of Rewriting Systems

Jean-Marc Steyaert (LIX, Ecole Polytechnique, France)
email: steyaert@lix.polytechnique.fr

It is quite often easier to present a program in terms of rewriting rules; one usually gets a more concise and more formalized version of the algorithm, which allows an easier approach to complexity analysis. We present general tools for analyzing the average case behaviour of rewriting systems working on algebraic or logical expressions. A method is proposed to obtain in a rather systematic way asymptotic estimates for several classes of programs that perform various kinds of simplification. This method, which has been partially automated, provides the programmer with a powerfull tool for a priori profiling and optimizing the algorithms during their conception.

Correctness of Data Representations in Algol-like Languages

Bob Tennant (Queen's Unversity, Ontario, Canada)

One of C.A.R. Hoare's most influential contributions to programming methodology and language design is a method described in 1972 for verifying data refinements. Although Hoare described the validity of this method as "a fairly obvious theorem", a rigorous proof for a realistic programming language has not been possible for the surprising reason that there has not been a satisfactory semantics of local-variable declarations in the context of higher-order procedures.

This talk will describe how this problem has been addressed by recent work by O'Hearn and Tennent which incorporates Hoare's ideas on data representation into the semantic interpretation.

Astracts of Contributed Talks

Resource-Oriented Model for Real-Time Systems

Clive Adams (University of York)
email:clive@minster.york.ac.uk

We present a new type of event structure - one that provides explicit support for asynchronous communication via named channels. By specifying each channel as either internal or external to

a target system we are able to define two notions of equivalence. The first, all-action equivalence, considers all actions of a system to be observable. The second, external equivalence, considers only the external communication actions of each system. This may be thought of as a more intuitive equivalence since it equates systems purely on the grounds of their externally observable behaviours. Finally we present a top-down refinement operator for our event structures that preserves these equivalences. We demonstrate that this operator is both well-defined and compositional.

The Contractum in Algebraic (Hyper)Graph Rewriting and its Applications

Richard Banach (University of Manchester)
email:banach@cs.man.ac.uk

Algebraic hypergraph rewriting proceeds by applying a rule $L \leftarrow K \rightarrow R$ to a redex $L \rightarrow G$. This entails first constructing the "pushout complement" D, of $K \rightarrow L \rightarrow G$ (a procedeure that in general fails if some "glueing" conditions do not hold), and second, the construction of the honest pushout of $D \leftarrow K \rightarrow R$; hence the alternative name "double pushout construction". By contrast, term graph rewriting proceeds by first pattern matching a redex $L \rightarrow G$, where L occurs as a subgraph of a larger term graph P. The matching permits the glueing into the redex of the remainder of P, "contractum building"; which is followed by the redirection phase to complete the rewrite. In many cases these two constructions have been shown to be equivalent, but nevertheless the double pushout analogue of contractum building has remained obscure. But this is easy. Take the pushout of the rule $L \leftarrow K \rightarrow R$ giving say $L \rightarrow P \leftarrow R$. The subgraph $P - L$ is the algebraic analogue of the TGR contractum. In fact the algebraic approach can be entirely reformulated in terms of $L \rightarrow P \leftarrow R$; the two approaches combine to give a "pushout cube" construction, the colimit of $K \rightarrow L$, $K \rightarrow D$, $K \rightarrow R$. The new form is useful for reasoning about acyclicity; if P is acyclic (in some useful sense), then so is the rewrite of any injective and short-circuit-free redex. If the redex is not injective or short-circuit-free, a type discipline may recover the preservation of acyclicity. These techniques give an alternative approach to correctness in interaction nets and multiplicative linear logic.

Formal Description of Distributed Multimedia Systems (TEMPO Project Summary)

G.S. Blair, H. Bowman, A.G. Chetwynd and L. Drayton (Lancaster University)
email:howie@uk.ac.lancs.comp

This talk will describe the Tempo project, which is a SERC/DTI funded project investigating the formal description of Distributed Multimedia Systems. The major motivation for such work is the development of formal techniques to be used in distributed systems standardisation (in particular, as required by ODP). The basic requirements for multimedia are support for continuous media and real-time synchronisation of these media tranmissions. The timing of media transmissions is maintained through Quality of Service parameters such as throughput, latency, and jitter.

During the project the suitability of existing timed formal techniques for specification of multimedia systems was considered. However, these approaches typically incorporate time directly into an existing notation. Such single language based approaches fail to reflect the functional/non-functional distinction central to the development of multimedia systems.

Tempo has thus considered a new approach in which behaviour (functional properties) is expressed in LOTOS and quality of service (non-functional properties) is expressed in the real-time temporal logic RTTL. The suitability of the new approach for expression of multimedia structures has been tested against a number of multimedia case studies, e.g. a multimedia stream and a lip synchronisation algorithm. We are at present considering the integration of LOTOS and RTTL, with a view to the development of validation techniques.

Finally, we further believe that dual language approaches are applicable to formal description of real-time systems in general. This is because such approaches tackle the fundamental conflict that exists between the necessity for specification of real-time to incorporate time and performance, and abstraction in formal methods.

Reducing the Size of Proofs And Program Developments in a Typed Lambda-Calculus

Alain Coste (Universite Paul Sabatier)

The Deva language, a higher order typed lambda-calculus, aims at formalizing and proving the development of programs from their specification. Their validity can then be mechanically checked. As wholly formalized developments are very heavy, methods helping the programmer's work must be found. This paper gives two partial answers to the problem: on one hand, modularity, which allows re-using already checked developments; on the other hand, the possibility to write partial developments, missing parts being synthesized by the validity checker itself.

After a succint presentation of the Deva language, we describe our two extensions taking into account the modularity and the automatic synthesis of expressions.

We have implemented an interpreter which takes into account the extensions proposed in this paper. It was first tested on developments written with the previous version of Deva, in order to guarantee an ascendant compatibility. Then, the developments were rewritten using the extensions, and checked with the new interpreter. A significant reduction of the developments size could thus be observed.

Upper Bounds for the Expected Length of a Longest Common Subsequence of Two Binary Sequences Vladimir Dancik and M. Paterson (University of Warwick)
email:vladimir@dcs.warwick.ac.uk

Let $f(n)$ be the expected length of a longest common subsequence of two random binary sequences of length n. It is known that there is a constant c such that $c = \lim_{n\to\infty} n^{-1}f(n)$ (Chvátal, Sankoff 1975). In this paper, a new method for obtaining upper bounds for this constant is given.

The many towers of Hanoi, a partial solution and its validation

Alan Dix (University of York)
email:alan@minster.york.ac.uk

Many years ago, in a lonely valley the foothills of the Himalayas, their lay two monasterys. In one of these were three crystal towers on which were stacked sixty-four golden rings of different sizes. Year by year the monks move the rings between the towers following the well known rules, and as is also well known when the task is finished the monks will sit back contented and the world disappear in a puff of smoke. Excitement mounted in the monastery, they had been at the task for four thousand years, and now over half of the disks had been moved. Recruitment was up, pilgrims and novices flocked to the monastery and across the valley their rival monastery looked on in despair. Then as the story goes, a wandering mathematician appeared and after scribbling on the back of a yak for a few moments declared that the world would be safe for another 450 billion years. Within days the pilgrims were dispersed and the towers fell into disrepute.

The abbot of the other monastery was not unduely alarmed at the demise of his rival, but saw that his own monastery was in need of a little publicity. If they should by chance find some towers of their own... But how many towers should there be? Too many towers would be a trifle expensive (have you ever tried to make a crystal tower), and besides if the task were to be finished in a few weeks and the world not end, what would people think? Yet if there were too few, and some inconvenient mathematician calculated the task would take millions of years to complete, it would lack popular appeal. What was needed was a task taking the odd thousand years, short enough to arouse interest, and long enough to hide away before it became a liability.

Programming in Type Theory

Andrew Douglas (University of Kent)
email:amd2@ukc.ac.uk

Type theory is the study of typed lambda calculi, and forms the basis of many programming languages. One in particular, the type theory of Per Martin-Löf, has long been seen as a programming language, as well as the logic of constructive mathematics.

We present here a syntax for a proposed higher order functional programming language, whose type system is based on Martin-Löf's type theory. The language allows primitive recursion and a primitive form of inductive types within a Miranda like language, and extends the type system to allow quantifiers which introduce bound variables into types, and an identity type.

Examples will show that this language can be seen to be both a constructive logic and a programming language, making it applicable to both program verification and specification.

Reversibility of 2D Cellular Automata

Bruno Durand (Laboratoire de l'Informatique du Parallélisme)
email:bdurand@lip.ens-lyon.fr

In this paper, we prove the co-NP-completeness of the following decision problem: "given a 2-dimensional cellular automaton $\mathcal{A}$, is $\mathcal{A}$ injective when restricted to periodic configurations of period not greater than its length?" In order to prove this result, we introduce a decision problem concerning tilings that we prove NP-complete. We then present a transformation of problems concerning tilings into problems concerning cellular automata.

In 1989, J. Kari proved that it is undecidable whether a 2-dimensional cellular automaton is reversible. Our complexity result shows that if the CA are restricted to bounded periodic configuration, although it is possible to decide their reversibility, this decision problem remains very difficult. Our result can be interesting for people who use reversible 2-dimensional CA as systems of public-key cryptography: a reversible CA is the key and, when applied on a picture placed on a torus, crypts it. The inverse CA decrypts. The key can be revealed to the public: nobody can construct the inverse cellular automaton with an efficient algorithm. Furthermore, the key can be very small and its inverse very large.

Associative-Commutative Matching via Bipartite Graph Matching

Steven Eker (Rutherford Appleton Laboratory)
email:steve@inf.rl.ac.uk

Term pattern matching modulo associative and commutative axioms (aka AC matching) is important in various term rewriting applications such as algebraic specification interpreters (OBJ3) and theorem provers (LP). The problem is known to be NP-complete complete in general, however for problem instances with linear patterns a polynomial time algorithm exists for finding a single matching substitution (if one exists). For finding all matching substitutions to a general problem instance, methods based on Diophantine equations, constraint propagation and exhaustive search have been proposed and implemented.

I will present a new algorithm which first converts a problem instance into a hierarchy of bipartite graph matching problems and then searches for consistent sets of solutions, using constraint propagation both to reduce the size of the graph hierarchy during construction and the size of the search space during the main search. For linear patterns it is possible to use a recently discovered enumeration algorithm to extract all matching substitutions with each substitution being computed in polynomial time.

Time as Probabilistc Behaviour

M. Fang, C.J. Ho-Stuart and H.S.M. Zedan (University of York)
{ming,cjhs,zedan}@minster.york.ac.uk

In this paper, we present a probabilistic process algebra for describing real-time, probabilistic behaviours in which 'time' is introduced by assuming the resolution of probabilistic choices to require one time unit. The development of this process algebra is based on a number of assumptions which we believe are important or desirable to model real-time probabilistic behaviours. Operational semantics and equivalence relations are also provided. We demonstrate that this model can be used to specify a large class of real-time, probabilistic systems.

An Abstract Approach To the Formal Design of Parameterised Synchronous Concurrent Algorithms (SCAs)

Matt Fairtlough (University of Sheffield)
email:matt@uk.ac.shef.dcs

In this talk I will report on an ongoing programme of research into the formal (i.e., machine-checked) design of SCAs.

The approach is abstract in two senses. Firstly, formal correctness proofs are specified over abstract datatypes, and secondly, the algorithms are expressed in the language PR which is itself the object of formal proof. Thus a formalisation of the meta-theory of PR has also been achieved.

The proofs are carried out using the LEGO proof development tool, which also supports the verification of parameterised families of SCAs.

A Simple $O(n \log n)$ Cost Parallel Algorithm for the Single Function Coarsest Partition Problem

Clive N. Galley and Costas S.Iliopoulos (Kings College)
email:clive@dcs.kcl.ac.uk

This paper shows a simple algorithm for solving the single function coarsest partition problem on the CRCW PRAM model of parallel computation using $O(n)$ processors in $O(\log n)$ time with $O(n^{1+\epsilon})$ space.

Solving the Reve's Problem Using Pascal's Triangle

David Gault (QUB Queen University, Belfast)
email:csg0860@v2.qub.ac.uk

The Reve's problem is a generalised form of the Towers of Hanoi problem in which more than 3 poles are allowed. A discussion of a technique for solving the puzzle and a minimal move solution for this technique is given. this solution may be derived using Pascal's Triangle. A proof of the optimality of dispersion and collection of discs will be given for this method. It remains conjectural that the particular technique always produces a solution in the minimal number of moves. It is known that there are other techniques which produce the same number of moves.

Real-time Specification and Refinement

Ian Hayes (University of Queensland)
email: ianh@cs.man.ac.uk

Our aim is to provide a top-level specification of a real-time system. Our specification should cover not only those parts of the system that are to be implemented in software but also those parts to be implemented in hardware. In fact, whether the system is implemented in hardware or software should not be a concern of the top-level specification.

To achieve our aims we make use of (topologically) continuous functions from time to the state of the system. These allow us to describe both discrete and analogue components in a consistent framework. To make the descriptions more readable we also use (physical) units rather than programming language types.

Our specifications are given in two parts: the assumption a process makes of its environment, and the effect it is required to achieve when placed in such an environment. These are analogous to pre- and post-conditions, but are really quite different as they do not involve a notion of before and after, rather the assumptions and effect describe properties over all time.

The use of such real-time specification statements allows for the development of a real-time refinement calculus similar to the sequential refinement calculi of Back, Morgan and Morris.

Towards a Proof Theory for Rewriting

Barney Hilken (Manchester University)
email:barney@cs.man.ac.uk

Proof theory is the study of logics not in terms of their consequence relations, but in terms of their proofs. The point of interest is not whether propositions are provable, but how they are proved, and what mathematical structure proofs have. In this talk we develop a proof theory for rewriting, not studying whether one term rewrites to another, but how.

Our approach is categorical: we see the terms and rewrites of a system as the objects and arrows of a category, analogously to Lambek and Scott's approach to logic. In this setting we define the basic notions of rewriting, including normal forms, confluence and normalisation. One new axiom on a category allows us to prove that strong normalisation implies the existence of normal forms and that confluence implies their uniqueness up to isomorphism.

As an example, we consider the simply typed lambda-calculus with beta- contraction and eta-expansion considered as the counit and unit of an adjunction respectively. We prove that the resulting system satisfies our definitions of confluence and strong normalisation, and we identify the normal forms.

The extra structure available in this approach gives strong normalisation in spite of the expanding direction of eta-conversion. The same technique could be used to solve many problems (such as commutativity and associativity) which are usually tackled by conditional rewriting.

Proof Sketches

Chris Holt (Unversity of Newcastle)
email:chris.holt@newcastle.ac.uk

Two great hurdles in convincing people of the value of program verification have been (i) difficulties in finding a notation suitable for reading and writing proofs, and (ii) the problem of size, since proofs tend to be an order of magnitude larger than either specifications or programs. One approach to resolving these problems is to use a single language for programming and verifying, in which a proof consists of a number of annotations to a program. The use of a visual language to convey structural information simplifies ifies annotations and so reduces the human burden; when combined with an intelligent checker, it has the potential to make proofs user-friendly.

Synchronous Concurrency and Dense Time

Chris Ho-Stuart (University of York)
email:cjhs@minster.york.ac.uk

Most real-time process algebras treat time synchronously, but handle simultaneous actions by interleaving. This implies that a behaviour of a process provides a temporal order on actions even when they occur in the same instant.

We suggest that it makes sense to regard all actions happening at the same instant as being strictly simultaneous, especially in a dense time domain. This idea is explored by providing new definitions for common algebraic connectives corresponding to synchrounous handling of actions. Actions are multisets of atomic actions, and simultaneous actions correspond to multiset union.

We take the language of timed CCS as defined by Liang Chen, and replace his prefix connective with simpler prefix connectives that separate action and time. A crucial feature of the new semantics is that an action prefix connective $a.P$ has the action a occuring simultaneously with P. Time is introduced with a bounded delay.

It would be misleading to call the resulting language timed SCCS, since actions have no inverses, and synchronous handling of the prefix implies the identity $a.b.P = b.a.P$. An action a can only occur *before* b is there is a time delay between them.

Weak bisimulation is a congruence for the resulting language, and an axiomization can be given which is complete for finite terms. A unique fix-point property is also provided for proving the equivalence of suitably guarded recursively defined terms.

On The Reuse of Additions in Matrix Multiplication

K. Kalorkoti (University of Edinburgh)
email:kk@dcs.ed.ac.uk

We consider the problem of multiplying pairs of matrices in terms of the reuse of additions. We show that if an algorithm is to be significantly faster than the naive matrix multiplication method then it must reuse additions to a great extent. (For example any bilinear algorithm for $n \times n$ matrix multiplication which does not reuse additions requires at least $n^3/8 - n^2/4$ arithmetic operations.)

Verifying Equivalences for Value-Passing Processes with VPAM

H.Lin (University of Sussex)
huimin@cogs.sussex.ac.uk

The standard approach to handle value-passing processes in process calculi like CCS and CSP is to translate them into "pure" calculi by introducing a choice operator over all values in the data domain involved. Such a translation introduces a source of infinity in to the language when the data domain is infinite, and as a consequence makes computer aided verification for value-passing process calculi impossible.

Recently semantic theories and proof systems for value-passing processes that avoid such translation have been proposed. The idea is to manipulate data *symbolically*, i.e. without instantiating input variables nor evaluating boolean/data expressions. In this talk we will describe a verification tool which implements such symbolic inference systems. It is called VPAM, for Value-passing Process Algebra Manipulator. Like its predecessor PAM, VPAM is an interactive proof assistant, not an automated tool; Like PAM, VPAM relies on rewriting for equational reasoning and induction for recursively defined processes; Also like PAM, VPAM allows the users to define their own calculi by specifying operators and axioms. But unlike PAM in which proofs are presented as a flattened collection of *sections*, VPAM supports goal-directed proof style. Proofs in VPAM are represented as *goal-trees* which grow in depth when proof rules are applied. In this way overall proof structures are made explicit so that more amenable to user manipulation. With the help of a window interface the tool is easy to use.

TCMC: A Novel Way of Compiling Functional Languages

Rafael D.Lins and Bruno O.Lira (University of Kent)
email:rdl@ukc.ac.uk

The traditional way to implement lazy functional languages was graph interpretation of combinators, as introduced by Turner. The understanding of the evaluation mechanisms of these languages allowed implementation to move from interpretation towards compilation, with substantialgain in performance.

Johnsson developed a strategy for compiling lazy functional languages, described as an abstract stack machine, called the G-Machine. The basic principle of the G-Machine is to avoid generating graphs. The code generated by the G-Machine when executed produces time and space performance at least an order of magnitude faster than interpreted functional languages. The original G-Machine implemented at Chalmers, Gothemburg, by Johnsson and his colleagues generated code in VAX-780 Assembly language, which made implementation extremely hard and machine dependant. It was common sense in the community of implementation of functional languages that assembly language implementation was the price to pay if one wanted efficiency. Chalmers LML compiler is still a reference in terms of performance of lazy functional languages. The G-machine way of controlling the execution flow and evaluation was followed by most of other implementations, even the ones based on different abstract machines as the Spineless G-Machine, the Spineless Tagless G-machine, TIM, and GM-C.

TCMC is a new abstract machine, in which the execution flow control is transferred to C, as much as possible. The key idea behind TCMC is to take advantage of efficient context switching in modern architectures based on RISC, which is able to implement function calls at a very low

cost. We also observed that the object code generated by C compilers is extremely neat and very fast. These factors lead us to try to translate each function definition into a procedure in C. It is obvious that not all scripts could be translated into C, if we wanted to have a lazy functional language. A higher-level abstract machine is still needed to glue together procedure calls, unevaluated expressions and functions, data-structures, etc.

Cyclic Weighted Reference Counting

Rafael D.Lins and Richard E.Jones (University of Kent)
email:rdl@ukc.ac.uk

Multiprocessor architectures are already a reality and open the possibility for the efficient solution of many problems in computer science. In distributed memory multiprocessors each processor is responsible for allocating and reclaiming structures residing in its local memory. Processors communicate amongst each other to maintain memory management. Inter-processor communication is far less efficient than local memory access. Avoiding communication is known to be a way of increasing efficiency in distributed systems.

Weighted reference counting is a very simple and efficient memory management system for multiprocessor architectures. However it is only able to deal with acyclic data structures. In this talk we present a weighted reference counting algorithm which is able to work with data structures.

A New Interpretation of the Structured Operational Semantics for Hiding Operator of Algebraic Processes

Wenbo Mao (Manchester University)
email:wenbo@uk.ac.man.ee.comms

For CSP processes (resp. Hennessy processes), failure equivalence (resp. testing equivalence has an exceptional advantage of reducibility. Namely, a process which is hierarchically constructed with the use of parallel and hiding operators can be semantically approximated by another process which is expressible by only using sequential operators (ie without need of resorting to the parallel and hiding operators). However desirable in supporting automated verification, it is known that substantial amounts of time and space are needed for computing the reduction.

A new interpretation of Plotkin's structured operational semantics for hiding operator is proposed in this work. Under the new interpretation an operational semantics is found to permit the notion of reducibility while without expanding the term size due to the operation of hiding. A considerable difference in computational complexity between the new semantics and that of failure will be paradigmatically demonstrated. Besides, the new semantics generates a consistent algebraic system with respect to its operational intuition.

Formal Logics for Secure Protocol Analysis

Wenbo Mao (Manchester University)
email:wenbo@uk.ac.man.ee.comms

A security protocol, such as one for distributing cryptographic keys, is essentially a few lines of a specification of a program. Its analysis can therefore be considered as analogous to the correctness verification of such a program. However, unlike the case of running a computer program, where the user naturally bears an intention to follow the instructions so to avoid potential bugs, the main objective of a dishonest user during a run of a security protocol is to exploit its bugs and through the abuse to obtain or alter vital information. The damage caused can be disastrous. Hence, debugging of a security protocol must be so thorough that its abuse can achieve nothing.

Such thorough debugging is possible only through formal approaches. The pioneering work of Burrows, Abadi and Needham (the BAN logic has been recognised to put forward a correct direction in the area of study. It is our understanding, however, issues such as interpretation of protocol semantics, soundness and completeness of protocol analysis are informally dealt with and thereby the logic permits approval of dangerous protocols. Measures to make the BAN logic formal will be discussed.

Algebras and the Approximation of Finite and Infinite Synchronous Concurrent Algorithms

Brian McConnell (Unversity of Edinburgh)
email:bm@dcs.ed.ac.uk

We present an algebraic model of finite and infinite Synchronous Concurrent Algorithms (SCAs). An SCA is a *parallel deterministic algorithm* consisting of a network of modules and channels, computing and communicating data in parallel, and synchronised by a global clock. Each module is a basic computational unit with finitely many input and output channels. An infinite SCA (ISCA) has infinitely many modules and channels. We will represent an SCA by a many sorted algebra.

We will then discuss the following. Given a family $\langle SCA_i \mid i \in I \rangle$ of algebras representing SCAs where the SCAs simulate or approximate one another with respect to some property, and this is coded by homomorphisms between the algebras. Suppose this simulation or approximation relation defines a directed partial order on the family. Such a family of SCA algebras forms a *direct system* and, in particular, if the family consists of subalgebras of an SCA algebra, and is ordered by inclusion, then the family forms a *local system*. Given an SCA represented by an algebra, we view approximations as subalgebras. We then consider the approximation of SCAs through the approximation of locations, input streams, initial states and finally data.

A CCS Model of SQL Queries

P.Murray (The University of Sheffield)
email:p.murray@uk.ac.shef.dcs

SQL is a commonly used database language with an international standard. Due to its popularity a lot of work has gone into transformation or translation of it. However, the correctness of such work is difficult to asses because, as with most programming languages, the SQL standard lacks a formal semantic definition.

A programming language is usually defined by its lexical, syntax and semantics rules. These three determine what tokens may appear in the language, in what order and what effect these will have. Often each individual token is given its own meaning or behaviour, which sometimes varies according to the context in which it is used (referred to as overloading). In the definition of SQL the lexical rules and the syntax are covered in one grammar expressed in EBNF. The semantics is a more complicated matter. The behaviour of each token is expressed in natural language. From the standard definition it is difficult to determine the exact behaviour of a query given the combined component definitions.

The process algebra CCS can be used to specify the semantics of a language so long as all the aspects of the programs behaviour are carefully covered. Important aspects include the behaviour represented by the tokens within their context; the sequence of these behavioural units (sequential or parallel); and the behaviour of any implicit objects such as data, communication channels, memory, etc. An agent can be defined for each token of the language, with different versions to cover the case of overloaded tokens, and a control mechanism to ensure correct sequencing. Further agents can be defined to represent the implicit objects. Finally a translation from the source language is used to define how the CCS agents are linked to form a model of the original program.

The above approach has been used to specify the semantics of SQL queries. An overview will be given of the CCS model including an examination of the structure of SQL queries; the control mechanism of the CCS model; the token agents; the data agents; and the behaviour of the model including termination. This approach to semantic specification creates the opportunity to compare SQL queries to other languages which may be specified in the same way.

Densely Embedding the Complete Binary Tree in Communication Networks

Somasundaram Ravindran and Alan Gibbons (University of Warwick)
email:ravi@dcs.warwick.ac.uk

We describe dense edge-disjoint embedding of the complete binary tree with n leaves in the shuffle-exchange and de Bruijn networks and in the hypercubes all with n nodes such that each network node plays host to exactly one leaf of the tree and at most one internal node of the tree. In all cases, the maximum embedded path length from a leaf to the root is optimally short. This facilitates the efficient implementation of many P-RAM algorithms on these architectures.

Approximating Minimum Weight Perfect Matchings in Complete Graphs with an $O(\log n)$ Performance Ratio is in NC

S. Ravindran, N.W. Holloway and A.M. Gibbons (University of Warwick)
email:ravi@dcs.warwick.ac.uk

The problem of finding efficient parallel algorithms for optimum matchings in graphs is a crucially important sub-task for many other problems. Indeed, it has been stated that whether seemingly simple problems like finding a minimum weight perfect matching in a complete graph (which is known to be in RNC) is in NC will determine a modern answer to the question "What is an efficient parallel algorithm?". Will the answer be "Those problems in NC" or "Those problems in RNC". The question of whether the problem of finding a minimum weight perfect matching is in NC remains open. However, we show that the problem can be approximated in NC with a logarithmic performance ratio.

Towards Task Sets: The TAM Real-Time Refinement Calculus

David J. Scholefield (University of York) djs@minster.york.ac.uk

Existing step-wise refinement calculi for the development of real-time systems do not cover the entire development spectrum; they either allow specifications to be refined to more concrete specifications, or implementations to be refined to machine code. We present a refinement calculus for the development of real-time programs in which specifications may be transformed into executable code. The calculus is called the Temporal Agent Model (TAM), and uses a wide-spectrum language in which temporal logic assertions coexist with more imperative statements such as assignments, concurrency, and variable declaration. The calculus contains a set of refinement laws which define how logic assertions may be replaced by semantically equivalent (or stronger) code. Real-time issues such as task periodicity, sporadic task handlers, deadlines, and access to a real-time clock, are handled explicitly in the language, thus enabling the developer to use existing scheduling theory when analysing the completed program. In this paper the language is defined, along with a number of refinement laws, and an example refinement case study is undertaken.

Term Rewriting Systems as Sesqui-Categories

John G. Stell (Keele University)
john@cs.keele.ac.uk

It has been known for some time that a term rewriting system can be modelled as a 2-category, where the objects are variable symbols, the morphisms, or 1-cells, are terms, and the 2-cells are rewrites between terms. In this talk I will show that a weaker structure, which I call a sesqui-category, may be used, and is able to capture some aspects of term rewriting which are lost in the 2-category model. A sesqui-category may be thought of as a 2-category without a general horizontal composition of 2-cells with 2-cells; there are however operations which compose 1-cells with 2-cells. One significant difference between a term rewriting system seen as 2-category, and as a sesqui-category, is that in the latter case the 2-cells support a notion of length, which corresponds exactly to the usual notion of length for rewrites. The possibility of applying the sesqui-category model to the study of confluence, and critical pairs, in term rewriting will be discussed.

Logical Description of Monotone NP Problems

Iain Stewart (University of Swansea)
email:csiain@pyr.swan.ac.uk

We introduce the class of problems NP_{C-RAT} accepted by polynonial time (non-deterministic) conjunctive random-access Turing machines (C-RATs): such mahcines crash when the oracle answers "no" (the oracle is always the input structure). We show that NP_{C-RAT} consists of all monotone problems in NP and we logically characterize this complexity class in two ways: the first involves an extension of first-order logic using an operator corresponding to some problem (an approach initiaed by Immerman); the second involves a restricted version of existential second-order logic (in the style of Fagin). In both logics, symbols belonging to the problem vocabulary are only allowed to occur positively. We also show that NP_{C-RAT} possesses complete problems for NP (via logspace reductions) are unlikely to be complete for NP via monotone projection transitions.

Cardinality of Strings Drawn from a Multiset

Brian R. Stonebridge (University of Bristol) email:brian@uk.ac.bristol.compsci

The problem which we consider is that of enumerating the set of all possible sequences consisting of the first $r \geq 1$ elements drawn from a multiset, $\mathbf{n}$, of $n \geq r$ elements, without replacements, and hence obtain a formula for the cardinality, $C(\mathbf{n}, r)$, of this set.
The method requires a summation over a set of bounded partitions, which is computationally intensive if tackled using generating functions.
We give an efficient method of calculating the set of distinct lexicographically ordered partitions of r, and hence a tractable method of evaluating the formula.
The result, which was motivated by a problem in the sequencing of requests in parallel processing systems, has wide applications in information theory and combinatorics.

Towards Libraries for Z

Luke Wildman (University of Queensland)
email:lwildman@cs.man.ac.uk

We consider adding parametrised libraries to Z as a strict extension to the current notation. We examine a simple modularisation facility with only generic sets as parameters, similar to current Z generic schemas.

We would like the modularisation facility to be usable with all forms of Z specifications, not just those making use of the conventions for specifying sequential programs (that is, specifications written in terms of a state and a number of state-to-state operations using conventions such as primed and unprimed variables, delta schemas, etc.). For this paper, we see a modularisation facility as a way of grouping together, and perhaps parametrising, a number of definitions, in the same way that a schema groups together components. A library can thus be viewed as a super-schema.

Finally, we consider interactions between the modularisation facility and current Z notation. In particular, we consider the problem of allowing flexibility in using free types and schema types as parameters.