REPORTS ON CONFERENCES

Report of the 10th British Colloquium for Theoretical Computer Science

University of Bristol, 28-30th March 1994
(Sponsors: SERC, Hewlett-Packard, Praxis, Sun microsystems)

The tenth meeting of the British Colloquium on Theoretical Computer Science was held in the of University of Bristol at the now traditional time close to the Easter festivities. The University and the historic city of Bristol provided excellent location for this celebratory conference which was admirably organised by Brian Stonebridge and his energetic local team. The colloquium was splendidly supported by nearly forty technical presentations (abstracts of contributed talks may be found below) including the stimulating contributions of five distinguished guests (Paul Spirakis, Robert Cori, David Bree, Richard Bird and Mark Jerrum).

BCTCS is now established as the foremost conference on theoretical computer science in the British conference calendar. Apart from providing a forum for the presentation and discussion of the current work of established researchers, the meeting has the primary aim of encouraging young researchers. To this end presentations are not subject to a refereeing process and young scientists benefit from the sympathetic and relaxed discussion of a wider and international community. This year saw the introduction of tutorial talks (Paul Dunne, *Complexity theory of Boolean functions* and Mike Holcombe, *X-machines - what are they?*) which will be a continuing feature of the colloquium. In addition, there are plans to provide financial assistance for the attendance of research students at future meetings. For the first time, this year's registration fee included one year's subscription to membership of the European Association for Theoretical Computer Science.

BCTCS11 will be held at the University College of Swansea, University of Wales over the inclusive dates 2-5th April 1995. The local organiser, Chris Tofts (e-address: c.m.n.Tofts@swansea.ac.uk or cmnt@cs.man.ac.uk) welcomes all enquiries from potential participants.

Alan Gibbons

BCTCS Local Organisers 1993: Brian Stonebridge (Chairman), Alan Chalmers, Neil Davies.
BCTCS National Committee: Alan Gibbons (Chairman, Warwick), Paul Dunne (Secretary, Liverpool), Iain Stewart (Treasurer, Swansea), Brian Stonebridge (Bristol), Julian Bradfield (Edinburgh), Zedan Hussein (York), Rick Thomas (Leicester), John Tucker (Swansea), Mike Holcombe (Sheffield).

Invited Talks

Paradigms for fast parallel approximations to problems that are hard to parallelise
Paul Spirakis, Patras, Greece

Building automata with time stamps
Robert Cori, Bordeaux and Paris, France

The semantics of natural language temporal prepositions and conjunctions
David Bree, Manchester, UK

Relational program derivation
Richard Bird, PRG Oxford

The computational complexity of counting
Mark Jerrum, Edinburgh, UK

Contributed Talks

On the Analysis of Genetic Algorithms

Martyn Amos, Somasundaram Ravindran and *Alan Gibbons*, University of Warwick

A Genetic Algorithm is an optimisation algorithm based on the theory of natural selection. Genetic Algorithms seem to be inherently highly parallelisable. Although empirical results apparently confirm this potential for fast parallel computation, there is little theory that allows a proper complexity analysis. We propose an analytical way forward through the application of Markov Chain theory.

Petri nets and modal logics

Julian Bradfield, University of Edinburgh

Suppose we are proving properties of infinite state systems, using some logic, such as the modal mu-calculus. We will wish to write down sets of states, and it would be nice to know the descriptive complexity of the sets we might need to write down. I show that although Petri nets are not Turing powerful, to prove arbitrary mu-calculus properties we still need to be able to write down undecidable sets of markings; indeed, we even have to write down non-arithmetic sets. (Reference: J. C. Bradfield, Verifying Temporal Properties of Systems. Birkhäuser, Boston, Massachussets (1991)).

Rewriting a Semigroup Presentation

C.M.Campbell, E.F.Robertson, N.Ruskuc and *R.M.Thomas**, University of Leicester

Let S be a finitely presented semigroup having a minimal left ideal L and a minimal right ideal R. We show how to obtain a presentation for the group $R \cap L$. It is obtained by rewriting the relations of S, using the actions of S on its minimal left and minimal right ideals. This allows the structure of the minimal two-sided ideal of S to be described explicitly in terms of a Rees matrix semigroup. These results are applied to the Fibonacci semigroups, proving the conjecture that $S(r, n, k)$ is infinite if g.c.d.$(n, k) > 1$ and g.c.d.$(n, r + k - 1) > 1$.

Modeling Process with Bounded Locations

Fabio Casablanca, Nagoya University

One of the aspects neglected by a interleaving approach to concurrency is the spatial distribution of the computation.

Approaches which have dealt so far with this problem assume the availability of an infinite number of sites where to perform actions or suppose that the spatial structure of a process is known before its execution.

We present a model where, coherently with the reality of concurrency, processes are extended with managers of finite sets of spatial resources (locations). The managers assign locations to agents, if any is available, and collect locations which are eventually released. Release of locations is modelled explicitly by a special release action.

Also, we suggest that managers can ask for and receive locations from other managers. For this purpose it is necessary to define a contiguity relation between managers and a location transfer policy. Alternative approaches to the modeling of these aspects are presented.

Finally the expressiveness of the model is discussed.

Automated Structural Test Data Generation

Michael Cousins , University of Portsmouth

The test generation problem is defined as "generate an input vector which forces execution flow down a predefined path through a program". We show how this problem can be transformed into an unconstrained optimisation. For real objective functions the optimisation can be performed using the simplex method of Nelder & Mead, the talk describes a method for extending the simplex algorithm to work with integer variables. Our own trials using the method suggest it is a promising approach - we have been able to successfully generate a test case for every feasible path we have tried giving a 100% cyclomatic coverage over all our trial programs. We conclude with an artificially generated example which highlights the limitations of our approach.

A Failures Model Applied to Divergence in Processes

S Counsell, Birkbeck College, London

In earlier work, we described a divergence model in terms of the ability of a process to deviate from an ideal path: we defined a non-divergent state as one matching a state on an ideal path, a partially-divergent state as a deviation from a state on the ideal path, and a fully-divergent state as a state within a never-ending loop. In this paper, we extend that model by attributing an action type to each action occurring between two states, whilst retaining the three divergent state types introduced in the previous model. This finer level of granularity allows us to characterise the behaviour of finite deterministic processes in terms of those action/state pairs in their *traces* and actions/state pairs *refused* which together form the *failures* of a process. By comparing the refusals of an arbitrrary process with those of a single ideal path within the ideal process, we are able to quantify the extent to which a process diverges from that ideal path, and hence order processes accordingly. The ordering is a partial-ordering, with maximal element the *ideal*, and a minimal element similar to the definition of *chaos*.

Upper bounds for the expected length of a longest common subsequence

Vlado Dančík, University of Warwick

Let $f(n)$ be the expected length of a longest common subsequence of two random sequences over a fixed alphabet of size k. It is known that $f(n) = c_k n$ for some constant c_k. We define a collation as a pair of sequences with marked matches. A dominated collation is a collation th at is not matched optimally. Upper bounds for c_k can be derived from upper bounds for the number of nondominated collations. Using local properties of matches we can eliminate many nondominated collations and improve upper bounds for c_k.

An Institution for Modular Specifications

E. David & C. Roques, Universited'Evry & Universited'Orsay

In this work we propose and discuss a comprehensive foundation to the notion of modular specifications.

Algebraic specifications provide a formal framework allowing to deal with vmodularity; the main goal of this work is to investigate modularity issues for algebraic specifications. We define a very general framework for modularity in the following way: first, we provide a generalisation of the (institution independant) stratified loose semantics. Then, we construct an institution from these modular specifications, and we show that some aspects of the institution framework do not suit very well with modularity issues.

A Graph-Based Approach To Resolution In Temporal Logic

Clare Dixon, University of Manchester

Proof procedures for classical logic such as tableaux and resolution have been used as a basis for the development of decision procedures for temporal logics. The application of the classical resolution rule fails in temporal logics as two complementary literals will not represent a complementary formula if they occur in different time contexts. Due to such problems with resolution the majority of decision procedures for temporal logics have been tableaux based.

In this talk, we present algorithms developed in order to implement a clausal resolution method for discrete, linear temporal logics, given in [Fis90]. As part of this method, temporal formulae are rewritten into a normal form and both 'non-temporal' and 'temporal' inference rules are applied. Non-temporal resolution is like classical resolution but is only applied to formulae which represent constraints applying to the same moment in time. The temporal resolution rule attempts to match conditions that must be eventually satisfied with sets of formulae that together imply that the condition will never be satisfied.

Through the use of a graph-based representation for the normal form, "efficient" search algorithms can be applied to detect sets of formulae for which temporal resolution is applicable. Further, rather than constructing the full graph structure, our algorithms only explore and construct as little of the graph as possible. These algorithms have been implemented and have been combined with sub-programs performing translation to normal form and non-temporal resolution

to produce an integrated resolution based temporal theorem-prover.
(Reference: [Fis91] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Sydney, Australia, August 1991. Morgan Kaufman.)

Ceilings of Monotone Boolean Functions

Paul E. Dunne, University of Liverpool

In this talk we introduce the concept of a monotone Boolean function, h, being a ceiling of another monotone Boolean function, g. The relationship 'h is a ceiling of g' is of interest since for any monotone Boolean function, f, there is a close relationship between the combinational and monotone network complexities of the function ((f AND g) OR h). In this talk we outline three groups of results concerning ceilings of monotone Boolean functions:

1. We describe necessary and sufficient conditions for h to be a ceiling of g.
2. We give a combinatorial estimate of the number of functions within a specific class that have 'non-trivial' ceilings.
3. We show how for any monotone Boolean function, g, a sequence of r functions { g(i) }, which includes g, may be constructed so that for all $i \geq 0$ g(i+1) is a ceiling of g(i) and for all Boolean functions f, if g(i) has polynomial monotone complexity (for all i) and f has superpolynomial combinational complexity then for at least one value of i the function (f AND g(i)) OR g(i+1)) has superpolynomial monotone complexity.

These results establish that the combinational to monotone translation given by Berkowitz' slice-functions are a special case of a more general property concerning monotone Boolean functions.

Performance Implications of Virtualization Of Massively Parallel Algorithm Implementation

C. A. Farrell† *C.S. Iliopoulos*† ‡ *D. H. Kieronska*† *M. Korda*‡
† Curtin University. ‡ King's College London, Strand. London

In this paper we investigate the accuracy of performance prediction for virtualised implementations of parallel algorithms on massively parallel SIMD architectures. Virtualisation is the process by which algorithms which assume n processors are implemented in a system with p processors, where $n > p$. Virtualisation is implemented in some form by any parallel environment that allows algorithms to assume more processors than are physically available on the machine.

Our results also show that some algorithms perform closer to their theoretically predicted performance than others. This work has implications for both algorithm designers and compiler writers since it provides insights into the effects of virtualisation on the efficiency of parallel algorithm implementations.

The Compositional Properties of Extensional Answers to Logic Queries

R. J. Gaizauskas, University of Sheffield

This covers results of my Ph.D. thesis (*Deriving Answers to Logical Queries*. Sussex 1992):

1. a semantic definition of 'extensional answer' for a first order query against a logic database of first order formulas can be provided which parallels that given for definite clauses in the logic programming tradition;
2. using these notions of answer for queries in both definite clause and first order logic, a number of compositionality results can be demonstrated which show how a complex query may be decomposed into components such that the answers to the complex query are a function of the answers to the components, themselves viewed as queries;
3. a notion of answer derivation, based on the first order compositionality results can be defined which is both sound and complete;

4. this notion of answer derivation provides the formal logical basis for a new approach to automated deductive question answering, which naturally lends itself to parallel computation.

On the lower bound for parallel string matching

Clive N. Galley, King's College London, University of London

We consider the problem of parallel string pattern matching over a general alphabet on the CRCW PRAM model of computation. Breslauer-Galil showed a lower bound $\Omega(\log\log m)$ on the time complexity of the above problem. Their bound holds for $m \geq 5.075 \cdot 10^{35}$. We improve this to $m \geq 2^{48}$.

ATLAS:A Typed Language for Algebraic Specification

B.M.Hearn, University College of Swansea.

We introduce the algebraic specification language ATLAS. This language allows an algebraic structure to be given to a specification at the level of both types and terms. ATLAS specifications are executed by type and term rewriting. The initial model semantics of type specifications and the equivalence of typed equational logic and type and term rewriting has been considered in Meinke [1991,1992].

Unifiable Algebras

Nick Holloway and *Alan Gibbons*, University of Warwick

For an algebra, $A = \{a_0, a_1, \ldots, a_k\}$, and a binary relation $\diamond$, we define A to be a *Unifiable Algebra* if, for any two expressions, each of which is an arbitrary parenthesisation of $v_1 \diamond v_2 \diamond \cdots \diamond v_n$, an instantiation of all variables v_i can be found such that the result of both expressions is the same.

We demonstrate the existence of certain classes of non-trivially unifiable algebras (e.g. which are non-associative algebras) which may be unified by sequential polynomial time algorithms or RNC parallel algorithms. For an arbitrary algebra, exponential time seems to be required to test for the unifiability of two expressions.

Applications of the notion of unifiable algebras include the discovery of polynomial time sequential algorithms and RNC parallel algorithms for non-trivial sub-classes of problems in mathematics and computer science which generally take exponential time. We briefly describe some applications.

Domains as Vector Spaces

Chris Holt, University of Newcastle

The foundations of mathematics are generally couched in terms of either logic/set theory or category theory. However, it seems that multi-sets (bags) arise frequently in computing science: so perhaps a mathematical foundation for computing might choose to not invoke the axiom of idempotence ($xx = x$). We can then distinguish the cases of a value occurring one or more times; and in fact we can introduce "anti-values", that join with values to yield the identity element (the empty bag). This corresponds to extending the domain of coefficients of values, first from 0,1 to the whole numbers, and then to the integers. Going further, coefficients can be extended to the reals, or the complex numbers, so that the domain forms a vector space. In such a space, ordinary mappings can be defined to be linear. This seems to lead to a different way of viewing domains; it is conjectured that it may simplify reasoning about program behaviour.

Functional Logic

Ian Holyer, University of Bristol

It is becoming ever more important to be able to express, test and prove program properties, in order to increase the level of confidence in the correctness of programs. However, programmers find these activities difficult; proof assistant packages tend to treat proof as a separate activity from programming, requiring different skills. Part of the problem is that the language of logic is less familiar and convenient than a programming language.

Functional languages, being the only purely declarative programming languages, provide a programming paradigm which is particularly clean and simple from a logical point of view. All

external aspects of program behaviour except time and space requirements can be deduced from the values of expressions.

In this talk, a case is made for using a functional language as its own logic language. Statements are boolean expressions within the language, and the usual boolean operators are used to combine statements. Some extended facilities are needed in the functional language to make it expressive enough to discuss properties such as program termination and the interactive behaviour of programs. The resulting functional logic has some of the properties of classical logic, and some of the properties of constructive (intuitionistic) logic.

The main advantage of this functional logic is that properties are computable; familiar programming techniques can be used in expressing properties, and they are automatically testable. In addition, programs do not have to be treated as text or syntax trees: program modules and property modules are just linked together.

X-machines with stacks

F. Ipate, University of Sheffield

We present the X-machine as a general abstract model of computation closely related to that of an automaton. The type Φ of a X-machine is the class of relations (usually partial functions) that constitute the elementary operations that the machine is capable of performing on the data structure X. By specifying the type Φ, many computational models can be derived, including finite state machines, pushdown automata, Turing machines.

The talk will be concentrated around two particular X-machines, stream X-machine and straight-move stream X-machine. A stream X-machine is one in which $X = \Gamma * x M x \Sigma *$, where M is the memory of the machine and Γ and Σ are the output and input alphabets respectively and any $\phi \epsilon \Phi$ will perform a sequential machine-like operation on $\Gamma * x \Sigma *$ (read an input character and produce an output character) while updating the value of M. A straight-move stream X-machine is a stream X-machine which admits empty moves (reads the null string and produces a null output). The class of relations and partial functions computed by these X-machine models will be investigated. These two models will be restricted to those in which the memory contain a finite number of stacks and Φ will include the usual stack specific operations and their computational power will be explored.

An Incisive Categorisation of Computable Database Queries

Mark Levene and *George Loizou*, University College London & Birkbeck College, London

We present an alternative approach to that of Chandra and Harel (*Computable queries for relational data bases*, Journal of Computer and System Sciences 21 (1980) 156-178) and Abiteboul and Vianu (*Procedural languages for database queries and updates*, Journal of Computer and System Sciences 41 (1990) 181-229) in considering computable database queries, which are mappings from sets of records to sets of records. In particular, we view a computable query as being realised via a Turing-computable mapping from strings to strings and an encoding, which encodes the input set of records into an appropriate string. An encoding of a set of records consists of two components: an ordering function, which orders the records in a set as well as the values of each record in the set, and an isomorphism, which maps the values in the records of the set to strings. An important class of encodings called free encodings, whose isomorphism has the same semantics as the identity mapping on record values, is also defined.

Our analysis of computable queries provides clarification of the notion of a computable query by dealing with the problem of how a database language can be implemented on a standard Turing machine that does not cater directly for mappings from sets of records to sets of records. We carry out our analysis by investigating subclasses of computable queries and showing the equivalence of these subclasses to ones already defined in the database literature. We also investigate an equivalence relation on computable queries; two computable database queries are related if they are realised via the same Turing-computable mapping, say f. We show the following interesting result regarding the cardinality of the equivalence class of a computable query with respect to the said equivalence relation: either f does not realise any computable query, or f realises exactly one computable query, or f realises a countably infinite set of computable queries. Our final

result shows that, by adding membership queries to the class of encoding-independent computable queries, the closure of the resulting extended class under composition of mappings is the set of all isomorphism-independent computable queries.

A Complexity Theory for Programs

Steve Matthews, University of Warwick

Complexity theory serves us well in providing a tool for scientifically comparing the efficiency of two algorithms. Any debate over the performance of an algorithm can be stated mathematically, for example, is it n*(log n) ? Consequently, we can use the language of mathematics to decisively answer such questions without resorting to personal opinion or prejudice.

Debates upon the comparative performance of programming languages do not appear to be conducted with anything approaching the same level of objectivity. Perhaps this is because there is no "complexity theory for programs", that is, no accepted scientific theory in which we can express the performance of a program.

In this talk we argue that it should be possible to ascribe the notion of "complexity" to a program by developing a theory of quantities for both the representations of data objects and the way they are manipulated.

On the complexity of deciding bisimulation equivalence of normed context-free processes

Faron Moller, Swedish Institute for Computer Science

In a recent volume of the Journal of Theoretical Computer Science, Huynh and Tian demonstrate that the problem of deciding bisimilarity between normed context-free processes is in $\Sigma_2^P = NP^{NP}$; that is, they present an algorithm which guesses a polynomial-sized proof of equivalence and validates this proof in polynomial time using oracles which freely answer questions which are in NP. In this talk we present a polynomial-time algorithm for solving this problem. As a corollary we improve on the singly exponential complexity bound for the language equivalence problem for simple grammars recently demonstrated by Caucal.

VLSI Circuitry for a Combined Paradigm Processor

Mark J. Neal, University of Wales

This paper considers some of the requirements and some possible structures for the VLSI inside a processor that contains both conventional and neural functionality. The VLSI neural hardware contains a degree of programmable flexibility and a feasible design is described in terms of components realisable with current technology. Having established the overall feasibility of such circuitry I then consider an outline design of a processor that combines conventional with neural processing. This type of device would provide an important resource for the construction of flexible neural network solutions to a wide variety of real-time and/or small embedded computer systems.

On the minimal realizations of a finite sequence

Graham Norton, University of Bristol

By a finite sequence, we mean a finite sequence of elements from a commutative domain R. We develop the theory of minimal realizations of a finite sequence *from first principles*. Our notion of a minimal realization generalizes the notions of a partial realization (in the sense of Kalman), of a Padé approximation and of a linear recurring sequence.

From this theory we derive an algorithm which computes a minimal realization of a sequence of length L in at most $L(5L+1)/2R$ multiplications. When R has unique factorization, our algorithm can be used to solve the partial realization problem, to decode cyclic codes (not only BCH and Reed-Solomon codes, but more general Goppa codes), to solve linear systems over R (extending Wiedemann's method), computing growth functions, the minimal polynomial of an R matrix, symbolic Padé approximations and of course computing the linear complexity of an R sequence. Thus we provide a common framework for solving some problems in Linear Systems (Control), Coding and Computing.

An Abstract Formalisation of MASCOT

Stephen Paynter, BAe Defence Ltd

MASCOT, the software design notation for concurrent and embedded programs used extensively within the UK defence industry, is briefly introduced. A case for using node labelled controlled graph grammars to define the abstract syntax of graphical notations is argued, and a simple 12 rule grammar is given which defines the abstract syntax of a large subset of MASCOT designs. The definition of a simple denotational semantic model for MASCOT is structured using this grammar. The model is defined using CSP.

Specifying software components for reuse

M Ramachandran, Liverpool John Moores University

Software reuse has been one of the hot research topic in software engineering. However, it has failed to keep its promises and has not address some of the fundamental research issues. One of the approach to reuse is library based components repository (Biggerstaff and Perlis 1989). We are unable to find a classification mechanism. There are a number issues relating to this approach:

1. How do we retrieve such components?
2. How do we specify what is required?
3. What do we classify our application domain?
4. What are the characteristics of a reusable components?

In this talk I illustrate how a simple semantic specification of a reusable component can be effective for reusing and retrieving from a repository. Our approach is to use simple formal specification technique which is object-oriented and allows you to specify the characteristics of a reusable component. It also allows you to compose components using a simple union and association. We are also interested in making the notation into more natural language keywords rather than complex notations. (Reference: Biggerstaff, T J. and Perlis, A J. (1989), Software reusability: Vol 1 and Vol 2, Addison-Wesley).

A New Approach To Algorithm Decomposition For Parallelism

Simon McIntosh-Smith, University of Wales, College of Cardiff

We present a system that assists in the design of parallel programs that are to be run on distributed memory multicomputers. The user provides a high-level algorithm description, which may or may not contain any explicitly parallel constructs. The system then provides intelligent assistance to help in the parallelisation process. It does this by identifying algorithmic features within the algorithm that can form parallel constructs either directly, or by applying some transformation. A main aim of the system is to produce readable parallel programs. A prototype system exists and initial results indicate that the system can provide useful assistance in the parallelising process.

Algebraic Specification with Transfinite Types: A Case Study of Evolving Algebras

L. J. Steggles, University College of Swansea

The theory of higher order algebra is a useful formalism for computer systems specification (see for example Möller [1987], Heering [1992], Meinke [1992] and the case study Meinke and Steggles [1994]). In this talk we show how to extend higher order algebra with *transfinite types*. These lead to more expressive algebraic structures which are well suited to modelling various forms of polymorphism, for example families of elements of varying type. For this class of algebraic structures useful algebraic properties such as initiality and a sound and complete equational calculus can be obtained.

We consider as a motivating case study the specification of *evolving algebras* (see Gurevich [1991]). An evolving algebra is an abstract machine in which a state is modelled by a single–sorted algebra and state transitions are defined by transition rules on algebras. They were originally

developed as a means of providing an operational semantics for programming languages, e.g. ISO standard Prolog (Börger and Daessler [1990]). We consider how evolving algebras can be algebraically specified using higher order algebra with transfinite types.

(References
E. Börger and K. Daessler. PROLOG.DIN papers for discussion, in ISO/IEC JTCISC22 WG17 N.58, National Physical Laboratory, Middlesex, page 114, 1990.
Y. Gurevich. Evolving algebras: a tutorial introduction. *EATCS Bulletin*, 43, pages 264–284, 1991.
J. Heering. Implementing higher-order algebraic specifications. In: D. Miller (ed), *Proceedings of the 1992 Workshop on the LambdaProlog Programming Language*. University of Pennsylvania, Philadelphia, 1992.
K. Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100:385–417, 1992.
K. Meinke and L. J. Steggles. Specification and verification in higher order algebra: a case study of convolution. To appear in: J. Heering, K. Meinke, B. Möller and T. Nipkow (eds), *Proceedings of HOA '93: An International Workshop on Higher Order Algebra, Logic and Term Rewriting*, LNCS Springer-Verlag, 1994.
B. Möller. Algebraic specifications with higher-order operators. In: L.G.L.T. Meertens (ed), *Program specification and transformation*. North Holland. Amsterdam. 1987.)

The Algebraic Specification of Programming Languages

K Stephenson, University College of Swansea,

We propose a method to formally describe programming languages by using *algebraic specifications*. An algebraic specification (Σ, E) consists of a set of equations E over the signature Σ. By expressing the equations as rewrite rules, the specification can be transformed into a term rewriting system (Σ, E). The initial model $I(\Sigma, E)$ of any such specification defines a Σ-algebra which will satisfy the equations E.

The relationship between context-free languages and closed term algebras has been well established. Our method extends this connection to the representation of non-context-free languages by algebraic specifications. These high-level descriptions give a natural, but formal definition of the syntax of programming languages.

We present a case study of a specification of a parallel language that illustrates our technique, and the advantages it has over traditional grammar theory descriptions.

The aim of this research is directed at equationally expressing the correctness of a compiler.

Informal Proof: A proof of Morley's theorem

Brian R. Stonebridge, University of Bristol

The notion of proof is pervasive, yet imprecise. The methodology, which starts from given axioms, and proceeds only by prescribed inference steps of a sound system of logic, is attractive. However, as has been pointed out by Alan Robinson (Syracuse), unless kept under control this may lead to a "proof" which is manageable only by a machine.

Since working theoreticians often wish to retain the hands-on feel for their activities, they often choose to adopt an informal approach. Although this relies on fundamentals, it may be difficult to formalise: but, if we can, we may be able to mechanise the development of informal proofs.

Morley's theorem states that the triangle formed by the points of intersection of the adjacent trisectors of the angles of any triangle is equilateral. Robinson pointed out that formal geometric proofs are known, yet they do not capture the essential symmetries of the problem. We provide an informal proof, by a construction.

An algebraic method for analysing responses to exercises

Paul Strickland, Liverpool John Moores University

The SLOTH system uses a rewrite-rule based approach to compare answers to student exercises in mathematics for software engineering. The aim is to allow course lecturers to specify their own chosen syntax and questions, while being able to analyse common mistakes and alternative syntaxes. This allows both immediate feedback in the tutorial environment, and also the possiblity to specify detailed marking schemes for formal assessments.

Enhancing temporal semantics in the database environment to support decision making

Yuan Sun, Dilip Patel, Denise Webster and *Paul Schleifer*
School of Computing IT and Maths. South Bank University London

The goal of this work is to create a decision support system by utilising temporal semantics. The research thereby intends to achieve the following goals:

1) To provide a modelling methodology which integrates data structure and behaviour for active temporal data.

2) To develop a representation for semantics of temporal information which will generate multiple versions of historical information, and to develop a summary mechanism.

3) To improve database performance by increasing the degree of data sharing, and to construct a structure for version specification of summarised historical data.

4) To present the query results both in tabular and graphical form by a specifically designed query language or interface which will provide a platform to accommodate decision support utilities, such as interpolation tools, summary specification, statistical functions and so on.

The applications for such work can be found in safety critical systems, airline reservation systems and business planning systems.

Verifying Functional Programs

Simon Thompson, University of Kent at Canterbury

Functional languages have the reputation of making program verification simple and elegant. This talk will explore this proposition. In particular, it will give - a design of a logic of the Miranda programming language; - an overview of two projects at the University of Kent to implement the logic: one using a system built in Miranda, the other using Isabelle.

The Importance of Theoretical Computer Science - a view from industry.

Martyn Thomas, Chairman and Founder of Praxis

Theoretical computer science suffers from two disadvantages: the argument over whether there is such a subject (and if there is, what its corpus of knowledge contains); and the widely held view that theoretical computer science is too distant from the industrial use of computers to be interesting. As a result, computer science departments have come under pressure to tackle problems that are closer to the market, to demonstrate that their research is worth spending taxpayers money on. Often, such research projects turn out to be little more than tool-building, and the tools are rarely of industrial strength and never get widely used.

Yet industrial computing faces huge problems, as shown by project overruns, failures and the occasional catastrophe. We in industry are tackling large-scale engineering problems, often without suitable processes, methods, tools or components.

In this talk, Martyn Thomas offers a view of the relevance of theoretical computer science to some of the problems faced by industry.