- F.-J. Brandenburg (Passau), M. Chytil (Prague): Syntax-directed Translation of Picture Words
- P. Degano, U. Montanari (Pisa): Chemical Machine via Graph Rewriting
- A. Paz (Haifa): On the Theory of Graphoids: a Survey

System Presentations:

- M. Himsolt (Passau): GRAPH^{ED}: An Interactive Tool for Developing Graph Grammars
- M. von der Beeck, A. Schürr (Aachen): PROGESS Editor: A Text Oriented Hybrid Editoer for PROgrammed Graph REwriting SyStems
- M. von der Beeck, A. Schürr (Aachen): IPSEN Environment: An Integrated and Incremental Project Support Evironment
- R. Sleep (Norwich): DACTL
- E. Wankle (Paderborn): PLEXUS: Tools for Analyzing Graph Grammars

---

Sixth British Colloquium for Theoretical Computer Science

Manchester, March 1990

Supported by SERC (Logfit) and IBM (UK)

Abstracts of invited and contributed talks

Organising Committee: John Tucker (Swansea), Mark Jerrum (Edinburgh), John Shawe-Taylor (RHBNC), David Rydeheard (Manchester), Paul Dunne (Liverpool), (Swansea), Alan Gibbons (Warwick), Harold Simmons (Aberdeen), Mike Paterson (Warwick), Ben Thompson (Swansea).

Local Committee: Sean Bechhofer, Carmel Dickinson, John Fitzgerald, Graham Gough and David Rydeheard.

Next meeting: will take place in Liverpool on March 25-28, 1991 and is to be organised by Paul Dunne (`ped@mva.cs.liv.ac.uk`).

A survey of Proof Systems dealing with Aliasing

Alain Ah-Kee
Praxis Systems PLC

Aliasing, which arises in procedural languages which permit parameter passing to procedures by reference, has typically been regarded as problematical for the semantic treatment of a language. The talk will give a survey of Hoare-like proof systems which attempt to deal with aliasing. The following systems will be surveyed:

- Reynold's specification logic: we shall look at proof rules for procedures and the way in which non-interference is specified.
- a number of proof systems (Cartwright and Oppen; Trakhtenbrot, Halpern and Meyer; Boehm) which deal with aliasing by explicitly distinguishing between values and locations.
- Brookes' proof system which uses an abstract sharing relation on identifiers to represent the notion of aliasing.
- Morgan's work on procedures which prohibits aliasing but is interesting in its treatment of procedure call and parameter passing by separate proof rules.
- the work of the author which deals with aliasing and the static scoping of program identifiers at the same time by introducing an environment in the system of proof rules, in a way akin to that usually used in denotational semantics.

Abstracts

Semantic Models for Term Graph Rewriting

Richard Banach
Department of Computer Science, University of Manchester

Term graphs are objects that locally look like terms, but globally resemble general directed graphs. They have been a vehicle for the implementation of functional languages for many years, but their use as a model of computation in its own right is of independent interest. DACTL is a relatively simple language which provides an operational semantics for the rewriting of term graphs. Predictably, the semantics is closer to rewriting theory than to direct implementation models that could be constructed and run efficiently on a distributed parallel machine. The issues that arise, concern depth of pattern matching, non-synchronisation of pattern matching, matching on nodes themselves undergoing rewriting, scheduling of rewriting primitives, and the atomicity and serialisability of the rewriting primitives. MONSTR is a simple sub-language of DACTL whose syntax may be given a number of different semantics which correspond to a greater or lesser extent to desirable implementation features. The relationship between these different models can be studied and a number of correctness results can be demonstrated. The main theoretical tools used in this endeavour are induction over the structure of executions, dataflow analyses of various kinds, and the analysis of various kinds of subgraph of the execution graph. All of these relate properties of the ruleset to invariants maintained by executions. The results obtained reinforce the intuition gained through practical experience, that for realistic problems, coded in "reasonable" ways, the semantic differences don't have a detrimental effect on correctness.

The Future of Imperative Logic

Howard Barringer (Invited speaker)
Department of Computer Science, University of Manchester

We distinguish two views of logic, the declarative and the imperative. The declarative view is the traditional one, and it manifests itself both syntactically and semantically. Syntactically a logical system is taken as being characterised by its set of theorems; it is not important how

these theorems are generated. Two different algorithmic systems generating the same set of theorems are considered as producing the same logic. Semantically a logic is considered as a set of formulae valid in all models.

In contrast to the above, the imperative view regards a logic syntactically as a dynamically generated set of theorems. Different generating systems may be considered as different logics. The way the theorems are generated are an integral part of the logic. From the semantic viewpoint, a logical formula is not *evaluated* in a model but performs *actions* on a model to get a *new* model. Formulae are accepted as valid according to what they do to models.

We explore this imperative view in the context of "executing" program specifications given by temporal formulae.

Towards a Formal Framework for Authentication

Colin Boyd
Department of Electrical Engineering, University of Manchester

Important recent work by Burrows, Abadi and Needham has defined a logic to analyse authentication protocols. At the same time, work has been proceeding in the International Standards Organisation (ISO) to develop a general framework for authentication in Open Systems. The work described in this paper is strongly influenced by both of these developments. A formal model for authentication is developed which encompasses the basic elements and is sufficiently general to describe a range of authentication protocols. The model has much scope for refinements and extensions.

The language used to describe the model is the Language for Temporal Ordering Specification (LOTOS) which has been standardised by ISO. LOTOS has a sound theoretical basis and is ideal in its expressiveness to describe communications protocols. The model describes processes which perform checking of the particular parameters required for authentication. These parameters are described via a formal abstract data type which can be made specific (*actualised* in the language of LOTOS) in the case of a particular protocol. Thus the model does not constitute a specification in itself, but it may be included in specifications of particular protocols, ensuring that a uniform and consistent notion of authentication is used. Examples of common authentication protocols are given and it is shown how they can be specified in such a manner as to include the model.

Proving Temporal Properties of Petri Nets

J.C. Bradfield
Department of Computer Science, University of Edinburgh

We present a sound and complete tableau system for proving temporal properties of Petri nets, expressed in the propositional μ-calculus. The system separates the checking of fix-points from the rest of the logic, so allows the use of net-specific analysis for this. We give examples demonstrating the use of the system on finite and infinite state systems.

Unification in Nilpotent Groups of Class 2 and 3

E.K. Burke
Department of Computer Studies, Leeds University

In this talk I will, firstly, consider unification for theories of nilpotent groups of class 2. I will outline the main method which consists of passing to the quotient by the commutator subgroup, applying a unification algorithm for abelian groups and lifting the solution to the free nilpotent group of class 2. There are certain difficulties concerning recognition of powers

which will be highlighted and discussed. For nilpotent groups of class 3 we follow a similar technique but use the algorithm for class 2 instead of the abelian algorithm. Related power recognition difficulties occur which will be compared with those for Class 2.

Comparing Algebraic Structures up to Algorithmic Equivalence

P. Byers
University of Surrey

Suppose one wished to compare two algebraic structures of the same signature by looking at the outcome of computations over those structures (in a programming language whose basic operations and tests are the functions and relations of the signature). In this paper we consider the notion of termination equivalence, which, if it holds between two structures, asserts that for every program P, P is total (i.e. it terminates for each choice of input variables) on one structure if it is total on the other structure. By choosing different programming languages, we obtain different instances of termination equivalence. We are especially interested in those instances which are congruences with respect to the datatype 'join' operation and show that termination equivalence for the language FAP is not a congruence. We also show that termination equivalence for the language FAPC is a congruence for effectively computable structures, and for these structures it is the smallest congruence containing termination equivalence for FAP.

Primitives for Virtual Shared Memory

Andrew Chin
Programming Research Group, Oxford University

PRAM-theory (shared memory) has long been the device by which parallel algorithms have been designed and analysed independent of machine topology. Recent work has focused on the costs of simulating the PRAM on realistic parallel architectures ("virtual shared memory"). In the algorithm designer's view, the memory resource for such a machine consists of local memory for each processor and a global memory. The physical connection network becomes a "black box".

The Block PRAM model was introduced last year by Aggarwal, Chandra and Snir as a complexity-theoretic model for virtual shared memory. Some fundamental parallel algorithms are more robust than others in this analysis. I will characterize and contrast the complexity of implementing prefix computations ("scans") and list ranking as primitives for the Block PRAM and describe applications to sorting and graph connectivity as well as the more general class of prefix-based algorithms.

Deriving complexity bounds from termination proofs

E.A. Cichon
Department of Computer Science, Royal Holloway and Bedford New College

Abstract Ordering Structures or Termination Orderings have been devised for proving termination of term rewriting systems. In their paper Proof-Theoretic Techniques for Term Rewriting Theory, Dershowitz and Okada calculated order types for some of the well known termination orderings. We show here that the order type of a termination ordering implies a complexity bound on the lengths of derivations. For example, in the case of a rewrite system where RPO (Recursive Path Ordering) proves termination, there is a primitive recursive function f and a constant c (which depend on the rewrite system) such that

$$\text{Length of longest derivation of term } t < f(\mathit{length}(t)).$$

String rewriting and homology of monoids

Daniel E. Cohen
Queen Mary and Westfield College

Squier proved that a monoid with a finite complete rewriting system must satisfy the condition FP_3 which arises in homology theory. Anick proved that such a monoid satisfies the stronger condition FP_∞. In this talk I give an account of this condition and some of its properties.

Traceable Automata with ε-moves

Duan Zhenhua
Department of Computer Science, University of Sheffield

In this paper, the concepts of Traceable Automata with ε-moves and ε-traces are introduced. The relationship among classes of languages accepted by some automata, deterministic traceable automata with ε-traces ($DTA_{\varepsilon\text{-}traces}$), determine traceable automata with nonfinal state ε-traces ($DTA_{nfs\text{-}\varepsilon\text{-}traces}$), and deterministic multi-phase traceable automata (DMTA), is proved to be as follows

$$L(DMTA) = L(DTA_{nfs\text{-}\varepsilon\text{-}traces})L(DTA_{\varepsilon\text{-}traces})$$

Formal Specifications for Rasterization Algorithms

S.M. Eker
Department of Computer Science, Royal Holloway and Bedford New College

Digitization (also called digitalization) is the process by which a continuous object (mathematical or 'real world') is converted in to a discrete form for input, output, or processing. In computer graphics, where the continuous object is a mathematically defined primitive (such as a curve or a polygon) and the discrete form is required for pixel or voxel based display systems the process is called rasterization or scan conversion.

Until recently rasterization algorithms have been designed and compared on an ad hoc basis with little thought given as to whether the object to be rasterized has some optimal discrete representation. More recent work by McIlroy, van Lierop and others has examined the properties of various rasterizations of lines and circles.

We develop a rasterization scheme for arbitrary objects in $\mathbf{R}^n$ based on a generalisation of Freeman's grid intersection scheme for 'real world' continuous curves. We determine necessary and sufficient conditions for an object to have a unique discrete form under this scheme and give more useful sufficient conditions for special classes of curves. We also examine the issues of discrete connectedness and intersection reduction (which are important from the point of view of designing efficient rasterization algorithms) with respect to our scheme.

Program and Proof Structures

Matthew V.H. Fairtlough
Department of Pure Mathematics, University of Leeds

We present a theoretical analysis of some classes of terminating programs. In particular, we compare recursive programs with while programs. To ensure termination, we use a supply of well-ordered, abstract computation structures as parameters to the recursions and while programs - these then determine the complexities of the corresponding computation trees, and thus of the underlying algorithms. The question now arises: given a set of recursive programs and a set of while programs which compute the same functions, how are the corresponding computation structures related ? Our analysis gives and approximately exponential relationship:

$$RECURSIVE(\omega.A) = WHILE(\omega^A).$$

We show how this relationship can be further analysed using traditional proof-theoretic techniques. The computation structures can also be viewed as abstract proof structures, and it transpires that the structures used for the computations are in fact those needed to prove termination of the relevant programs. We have:

$$PROVABLE(\omega.A) = RECURSIVE(\omega.A) = WHILE(2^{\omega.A}).$$

Some recent trends in the average case analysis of algorithms

Philippe Flajolet (Invited speaker)
INRIA

A large number of algorithms and data structures of fundamental use in computer science can be analyzed precisely in the average case. A classical paradigm decomposes an analysis into setting a collection of related combinatorial counting problems, finding the solutions to recurrence relations that express such counting problems, and last performing asymptotic evaluations. The method is especially fruitful when applied to problems of low (polynomial) computational complexity, for instance sorting and searching algorithms.

This talk will discuss recent mathematical approaches to problems of average case analysis in the domain of frequently encountered 'decomposable' data structures.

First, the emergence of symbolic methods in combinatorial counting avoids recurrences and permits us to derive directly generating function equations from specifications, for a large class of combinatorial counting problems. In this way, we can systematically compile specifications into functional equations of various forms.

Second, we know from classical analytic number theory that singularities of functions bear strong relations to the asymptotic form of their coefficients. A similar 'transfer' can be achieved here in many cases of interest for functional equations arising in the analysis of many computer algorithms.

Mathematically, this chain allows us to bypass the stage of explicit solutions to recurrences expressing average case analyses (we may not have closed forms), and it leads naturally to general and synthetic observations on the probabilistic properties of many classical combinatorial structures.

Computationally, this way of conducting analyses is also systematic enough. We will show how in the talk that it is possible to use the facilities offered by computer algebra systems

in order to develop an automatic analyzer that implements several of the combinatorial and asymptotic tools described above.

Tactics in programming within a proof framework

Didier Galmiche
CRIN Nancy

The aim is to present proof tactics in programming within a proof framework, more precisely in (second order) intuitionistic logic. The studies of the relationships between intuitionistic logic and computer science lead to the notion of programming with proofs which can be concisely summarized to: writing formulae as specification, proving this specification and extracting a program from this proof. An important point here is the correctness of the extracted programs.

But to derive interesting programs (efficient enough) we need to study the relationships between proofs and extracted programs and to have adequate proof tactics and strategies for program development in the framework. These tactics allow automatization of the proof construction process and we illustrate their applications with program development examples. Moreover, we precise connexions between data representation in the theory and the proofs and give some comparisons with another logical frameworks devoted to this approach.

Correctness-Oriented Development of a Specification Translator

Stephen Gilmore and Maurice Clint
University of Ulster and Queens University, Belfast

A software system is being developed which translates formal specifications expressed in the Z specification notation into equivalent specifications in the Meta-IV notation of the Vienna Development Method. Modules of the system are specified in a variety of specification notations and proofs of the correctness of the program code with respect to its specification are being produced as the implementation proceeds.

Unusually, correctness proofs are constructed entirely without machine assistance. The consequences of this bias are discussed. The influence of the requirement of proof construction on the design and implementation of the translator is also analysed.

The progress of the implementation is detailed and the difficulty of using several development approaches on a single project is analysed.

Theory of Graphical Expressions and Formal Notations

William Godwin
Gloucestershire CAT/Open

This talk outlines theoretical ideas being developed as a foundation for processing of formal notations such as are used in specification. A general framework is sought which treats diagrammatic forms on an equal standing with formulaic expressions.

The aim is to support linguistic operations on expressions, such as generation, editing, translation and transformation, within human limitations of complexity. These operations will form part of a prototype tool to aid design of formal graphical notations.

The theory builds upon and attempts to generalise current notions of Graph Grammar and Rewrite Rule, as applied to a range of structures which support notations. These notions are linked to a theory-building approach.

Category Theory and Topos Theory are proposed as appropriate tools for syntactic and semantic analysis; indeed diagrams are a useful vehicle for demonstration of categorical concepts in an accessible way.

Z notation is used both as a framework for constructing the theory and as a subject notation, exploring diagrammatic extension of Z formulae.

Zero Knowledge Interactive proofs

Shafi Goldwasser (Invited speaker)
MIT

This talk will survey zero knowledge interactive proofs in the single and multi-prove model.

Objects and Inheritance: An Algebraic View

Fiona Hayes
Hewlett Packard Laboratories

This talk relates object oriented programming to the ideas and techniques of algebraic specification. The basis of object orientated programming is modularity, encapsulation, and inheritance. Algebraic specification provides a clean framework for understanding these concepts.

In this talk, I will first define a model of data encapsulation based on observational equivalence. The different types of inheritance will then be defined using standard algebraic constructs.

The Complexity of the Cycle Index Polynomial

Leslie Ann Henderson
University of Edinburgh

In this talk, I investigate the computational difficulty of evaluating and approximately evaluating the *cycle index polynomial.* This polynomial is important in the field of combinatorial enumeration because it can be used to count structures up to isomorphism under a specified group of symmetries. The cycle index polynomial of a degree n permutation group is simply a weighted sum over the permutations in the group. The weights of the individual permutations depend upon the point $x \in \mathbf{R}^n$ at which the polynomial is evaluated. Since the size of a permutation group may be exponential in the size of its smallest generating set, it is a non-trivial problem to write an efficient algorithm which takes as input a set of generators for a permutation group and evaluates the corresponding cycle index polynomial at some fixed point x. In this talk I will show that this problem is computationally difficult for (almost) every point x. I will also show that at many points x it is computationally difficult to *approximate* the value of the cycle index polynomial.

Optimal Algorithms for Computing the Canonical Form of a Circular String

C.S. Iliopoulos (speaker) and W.E. Smyth
Royal Holloway and Bedford New College and McMaster University

An $O(logn)$ time CRCW PRAM algorithm for computing the least lexicographic rotation of a circular string (of length n) over a fixed alphabet is presented here. The logarithmic running time is achieved by using $O(n/logn)$ processors and its space complexity is linear. A second algorithm for unbounded alphabets requires $O(lognloglogn)$ units of time, using $O(n/logn)$ processors.

Functional Programming with a Visual Language

Chris Holt
Computing Lab., University of Newcastle

Visual languages have traditionally been developed as extensions of CASE tools and database query languages, with regions denoting entities/objects, and arcs denoting relations or control flow. This approach does not lend itself well to the description of simple, mathematically tractable programs.

An alternative is to develop visual paradigms for the basic mathematics underlying functional programming, that can be extended as more sophisticated structures are incorporated into the language. Visual complexity is then more clearly correlated with mathematical complexity (though of course shorthands can be developed for ease of use).

The language viz/f has been designed in this way. Arcs (lines) denote values, and regions denote relations (functions) over values. Primitive operations have been introduced that allow the general manipulation of elements of an arbitrary many-sorted algebra, within the framework of a three-valued logic. The introduction of quantifiers permits the description of specifications.

Compositional characterization of observable program properties

C. Barry Jay (speaker), Bernhard Steffen, Michael Mendler
LFCS, Edinburgh

In this paper we model both program behaviours and abstractions between them as lax functors, which generalize abstract interpretations by exploiting the natural ordering of program properties. This generalization provides a framework in which correctness and completeness of abstract interpretations naturally arize from the order. Furthermore, it supports modular and stepwise refinement: given a program behaviour, its characterization, which is a "best" correct and complete denotational semantics for it, can be determined in a compositional way.

A semantics for shared-state concurrency

Alan Jeffrey
Oxford

A synchronous process algebra is presented, and is given a transition-system semantics. Using this, we can give a transition-system semantics for a guarded command language with parallelism and local variables, where communication is by shared state.

Polynomial-time approximation algorithms for the Ising model

Mark Jerrum and Alistair Sinclair (speaker)
Edinburgh

The field of statistical physics is a rich source of examples and inspiration for the computer scientist. Of particular interest is the *Ising model*, which was proposed in the 1920s as a model for the behaviour of a ferromagnet and has generated a vast body of literature. The Ising model has since become a powerful paradigm for the investigation of more general cooperative systems in which short-range interactions between elements can give rise to long-range order. Empirical evidence suggests that many problems associated with Ising model, if treated *exactly*, are computationally intractable. This observation motivates a search for efficient *approximate* solutions.

The talk opens with an introduction to the Ising model and its associated computational problems. Then follows a discussion of recent work on approximation algorithms for the Ising model. These algorithms involve simulating a suitably defined stochastic process, and are distinguished from earlier approximation algorithms by having rigorously derived performance guarantees.

On the Metric Space of Traces

Marta Kwiatkowska
Department of Computing Studies,University of Leicester

An ultrametric for the domain of Mazurkiewicz's traces, which specializes to the usual metric for strings, has been defined. The metric space of traces is shown to be complete, and its topological properties are briefly discussed. The work allows to extend the metric approach to non-interleaving concurrency.

Systolic design and systolising compilation

Christian Lengauer (Invited speaker)
Department of Computer Science,University of Edinburgh

A *systolic array* is a distributed processor network with a particularly regular structure that can process large amounts of data quickly by accepting streams of inputs and producing streams of outputs. Application domains are numerical analysis, image or signal processing, text and speech processing, robotics, meteorology and others.

The regularity of the systolic array enables an automated synthesis from a more abstract description which does not address the issues of communication or concurrency. This automated synthesis is called *systolic design*; it leads to an abstract description of a systolic array.

In the past, systolic arrays have mostly been realized in hardware, but they can also be realized in software - in fact, this can provide a convenient and powerful way of programming distributed computers.

The talk presents a scheme for the transformation of traditional imperative programs into systolic programs. This process is called *systolising compilation*. The various techniques and formal methods involved will be demonstrated by way of a non-trivial yet comprehensible example: Gauss-Jordan elimination.

A systolising compilation is composed of steps that are performed in part automatically and in part by hand. At its core is the automatic infusion of optimal parallelism and communication.

What can you do with an equational reasoning theorem prover?

Ursula Martin (Invited speaker)
Department of Computer Science, Royal Holloway and Bedford New College

In this talk we give an introduction to some of the main ideas of automating equational reasoning by rewriting. The Squiggol style of progam development is shown to be readily automated using LP, an equational reasoning theorem prover. Higher-order functions are handled by currying and the introduction of an application operator. We present an automated verification of Bird's development of the maximum segment sum algorithm as an example of the power of this approach.

Can Functional Programming be Intensional?

Stephen G. Matthews
Department of Computer Science,University of Warwick

Functional Programming relies almost exclusively for its logic of programs upon the theory of recursive functions. An obvious implication of this is that graph reduction has a virtual stranglehold on the implementation of functional languages. Functional languages having temporal variables allow alternative techniques such as dataflow to be introduced, e.g. Lucid. In Lucid the marriage between recursion and temporality is not always a happy one.

The talk will examine the scope for using intensional ideas such as temporal logic in functional programming. We argue that if the marriage is not to end in divorce we must give an intensional interpretation to functional concepts such as application and substitution.

Development of Concurrent Programs in a Proof-Theoretical Framework: A Small Exercise

Dominique Mery
CNRS-LORIA-CRIN

The aim of our work is to illustrate a general framework, namely NU. This framework allows to derive proofs for concurrent programs from specification and in a step by step way. The relation between proofs and programs is defined in the concurrency framework.

An example, namely the garbage collection problem, is extensively presented and possible bugs of the suggested solutions are pointed out. NU manages invariance and eventuality properties according to a fairness assumption. The different steps of the development are expressed using a diagram based formalism. Connections with other frameworks are explained and further investigations are sketched.

Static Semantics using Inductive Invariants upon Augmented Grammars

Brian Monahan
Department of Computer Science, University of Manchester

Most formal term languages that arise in computer science, artificial intelligence and logic, can be specified as a finitely generated collection of ground term structures together with additional logical conditions for restricting the sentential phrases (i.e. instances) of the language. This restriction will often be expressed as an inductively defined predicate but with additional parameters containing context dependent information. Such a predicate is known as a *well-formedness condition* for the formal language.

A technique, called Augmented Grammars, is presented for describing well-formedness conditions more directly as an intrinsic constraint on the formation of instances in the language. It is also shown how the (recursive) type invariants that arise in specification languages such as VDM provides a ready-made notation for applying this as a practical specification technique.

Finally, a number of simple examples are given to illustrate how the technique can be applied in practice.

Time, Concurrency and Sheaves

David Murphy
University of Glasgow

Kwiatkowska, during the last BCTCS, presented a topological classification of properties of concurrent systems. Given a set of behaviours of a system, and a partial order indicating how one behaviour can be extended to another, various classes of properties (such as progress or safety properties) were identified. In particular, various common topologies erected over the poset of behaviours (such as the Alexandroff or Scott topologies), give various notions of 'open set', and each class of opens corresponds with a class of properties: a progress property, for instance, is just an Alexandroff-open set.

Here we extend this work in two directions. Firstly, time is added. A given class of behaviours is modelled using a sheaf over a model of time with the 'right' topology; real-timed progress properties, for instance, can be described using a sheaf erected from $\mathbf{R}$ with the Alexandroff topology to a poset of behaviours.

Secondly, we formulate Kwiatkowska's work in a category-theoretic setting, and so explore a more general notion of behaviour. Our sheaf-theoretic formalism is exploited to give a better understanding of the relationship between timing, behaviour and causality.

Finding a basis for the characteristic ideal of an n-dimensional linear recurring sequence

G.H. Norton (speaker) and P. Fitzpatrick
University of Bristol and University of Cork

We consider an n-dimensional linear recurring sequence (σ) of elements from a field $\mathbf{F}$, for $n \geq 1$. By Hilbert's basis theorem, the ideal $I(\sigma)$ of characteristic polynomials of (σ) is finitely generated. We present an algorithm IDEALBASE which determines a basis for $I(\sigma)$ under certain reasonable conditions. Our analysis applies in particular to doubly periodic arrays and to polynomial codes in several variables.

PAC Learning and the Vapnik-Chervonenkis Dimension

J.S. Shawe-Taylor, Martin Anthony and N.L. Biggs
Dept of Computer Science, Royal Holloway and Bedford New College, Dept of Maths, RHBNC and Dept of Maths, LSE

The concept of *Probably Approximately Correct* (PAC) learning was introduced by Valiant in 1984. Recent results have shown that a key condition for learnability is the Vapnik-Chervonenkis dimension of the hypothesis space. Assuming a finite dimension allows a bound to be placed on the sample size for which PAC learning can be guaranteed independently of the probability distribution on the input space. For hypothesis spaces with infinite VC dimension, however, no function is learnable in general, though learnability is possible for particular distributions. The talk will introduce the main concepts and report recent results, including improved bounds on the sample size and examples of infinite dimensional spaces where for particular distributions learning is possible.

Relating Concepts of Order-Sorted Theory

John G. Stell
Department of Mathematics and Computer Science, University of Keele

This talk presents a framework which clarifies the relationship between some of the various concepts of order-sorted theory and algebra.

The work originates in a programme to extend the existing category theoretic description of unsorted unification and term rewriting to the order-sorted case. This description represents substitutions by morphisms in a 'substitution category'. To define the appropriate category for order-sorted substitutions we are led to consider the notion of order-sorted theory as monad. This turns out to be a useful perspective from which to study the relationship between various concepts of order-sorted theory and algebra. In particular we obtain a precise description of the connection between two superficially different ways of presenting an order-sorted theory which have appeared in the literature on order-sorted unification.

Complete Problems involving Free Groups

Iain A. Stewart
Computing Laboratory, University of Newcastle upon Tyne

In this talk, we consider restrictions of the generalized word problem on finitely-generated subgroups of countably-generated free groups. In particular, we show that the generalized word problem on finitely-generated subgroups of countably-generated free groups is complete for **P** (via logspace reductions), but when we consider this problem where all words are of length 2, then the restricted problem is complete for **NSYMLOG**. While this is a new result, our methods of proof are more important in that we use Immerman's characterization of symmetric logspace as those problems expressible using the logic (FO + posSTC): up until now, the only way of showing that problems are in **NSYMLOG** was to use a complicated result due to Lewis and Papadimitriou. We believe that the results of this paper should encourage the capturing of complexity classes via logic. We also point out and remedy a discrepancy in a paper of Avenhaus and Madlener concerning problems involving finitely-generated free groups, and finally present some open problems.

A Poor Man's Synthetic Domain Theory

Paul Taylor
Imperial College

Synthetic domain theory is based on the slogan "Domains are sets, and all functions are computable". This has been studied for the effective (Hyland) topos by Wes Phoa. This paper is intended as a bridge between traditional ("Scott") domain theory and Phoa's work, by showing how many of his ideas can be implemented using the Yoneda embedding of a category (of domains) in a (Grothendieck) topos (the first part will be a "tutorial" on these traditional but ill-explained topics). Finally, the example of the Plotkin power domain will be considered.

A temporal calculus of communicating systems

Chris Tofts and Faron Moller
LFCS, Edinburgh and School of Biological Sciences, Bath

We introduce a calculus of communicating systems, an extension to that of *Timing Concurrent Processes, LFCS-89-103*, which allows for the expression and analysis of timing constraints, for example as in important for real-time processes. We present the language, along with its formal semantics and derive algebraic laws for reasoning about processes in the language. Though the core language is simple, we show that the language has several powerful derived operators which we demonstrate to be useful in several examples.

Three more Church-Turing Theses

J.V. Tucker
University College, Swansea

I will describe recent results on the classification of parallel deterministic computable functions; the specifiable functions; and the formally verifiable functions. If time allows I will discuss their application to the general theory of synchronous concurrent algorithms.

**From proof theory to proof search:
some remarks on the design of proof procedures**

Lincoln Wallen (Invited speaker)
Oxford

Herbrand's theorem is the major organising principle for machine-oriented methods of proof-search in the (classical) predicate calculus, but it fails to hold for almost all interesting non-classical systems. Gentzen's Midsequent theorem for prenex formulae has Herbrand's theorem as a corollary but, unlike the latter, does not extend to non-prenex formulae let alone non-classical systems in general.

We reformulate Herbrand's theorem using Gentzen methods to reveal the role of "permutability theorems" (in the sense of Kleene and Curry) relating to the conditions under which derivations are the same up to permutation of inferences. The key to resolution methods of proof-search is shown to be a full permutability theorem, enjoyed by (classical) propositional logic, but not by the predicate calculus. The occurs-check of unification is the means by which

the combinatorial advantages of full permutability are feasibly extended to the predicate calculus.

The reformulated Herbrand theorem, called a matrix theorem, does extend to intuitionistic and (normal) modal systems (propositional and first-order versions). With it comes the combinatorics of resolution and an insight into the complexity of the decision problems in terms of bounds on contraction.

Simplicity and Power: A New Theory of Computing

J.G. Wolff
School of Electronic Engineering Science, University of Wales, Bangor

In this talk I present a new theory of computing, called 'SP', and describe a new kind of computing system, under development, which is based on the theory.

The central conjecture in the SP theory is that all kinds of computing and formal reasoning may be seen as a search for *efficiency* in information where efficiency has a precise meaning in terms of Shannon's information theory and Chaitin's 'algorithmic information theory'.

Achieving efficiency in information means a search for *simplicity* in information without loss of usefulness of *power*. Achieving simplicity without loss of power means removing unnecessary repetition or *redundancy* in information. This in turn means the comparison or *matching* of patterns and the merging or *unification* of patterns which are the same – coupled with a process of *searching* the space of alternative unifications for those unifications which give the best overall reduction in redundancy.

The SP system will be designed for powerful searching of information to find efficient unifications of patterns, performing many operations in parallel.

The SP concept can integrate, rationalise and simplify a wide variety of computing functions. The SP system provides one simple language with many applications. Simplification and rationalisation of computing functions can mean substantial savings in training costs and in the costs of developing new applications.

The potential benefits of the SP benefits of the SP system can be seen in: the *management of knowledge* (organization of knowledge in expert systems and databases: information retrieval), *software engineering* (formal specification of software; object-oriented design; numerical computing; software re-use; configuration management; integrated project support environments) and *artificial intelligence* (inductive learning and automatic normalisation of knowledge structures; logic and logical inference; reasoning with knowledge which is incomplete or uncertain; natural language understanding, production and translation; pattern recognition; representation of plans and automatic planning). Some of these potential applications and benefits will be described.

Faster Circuits and Shorter Formulae for Arithmetic

Mike Paterson and Uri Zwik (speaker)
University of Warwick

Circuits and formulae over several bases, with improved depth and size respectively, are given for the carry save addition of n binary numbers (i.e., obtaining two numbers whose sum is equal to the sum of the n numbers). As a consequence improved circuits and formulae for multiple addition, multiplication and for most symmetric Boolean functions are obtained.
