REPORTS ON CONFERENCES

REPORT OF THE FIFTH BRITISH COLLOQUIUM FOR THEORETICAL COMPUTER SCIENCE (BCTCS5)

ROYAL HOLLOWAY AND BEDFORD NEW COLLEGE, LONDON
11TH-13TH APRIL 1989

Close to Easter, BCTCS5 was held this year at London University's countryside campus of Royal Holloway and Bedford New College. The founder's building of Thomas Holloway, red-bricked, white-stoned Victorian gothic, provided a magnificent setting for the conference amidst one hundred acres of wooded Surrey countryside. The local organising committee of Costas Iliopoulos, Alan Davies and John Shawe-Taylor provided warm and generous hospitality. The local arrangements ran so well that the usual causes of high anxiety for any organiser, such as the deeply-sleeping-in jet-lagged invited speaker and such as the setting off of the elaborate alarm system of the College's Picture Gallery on the occasion of the conference dinner, merely provided light entertainment for the assembled company. Mark Jerrum's informal speech of thanks at the conference dinner echoed everyone's appreciation of the hard-working organisers' success in providing a relaxed and happy conference.

There was a record number of more than forty technical presentations at the conference. Of these the following were delivered by the distinguished group of invited speakers. ***Parallel Algorithms on Words*** by Alberto Apostolico, ***Making formalism work for us*** by Roland C. Backhouse, ***A promising approach to complexity measures for computation over splay tree conjectures - recent advances*** byRichard Cole, ***Computations in Galois Theory*** by John Dixon,***Object Oriented Programming as a natural extension of functional programming*** by Joseph A. Goguen, ***The Coppo-Denzani Type System*** by Roger Hindley, ***Current Theoretical Issues in Logic Programming*** by John Lloyd and ***The Thesis that Computable Functions are Uniformly Continuous*** by Michael B. Smyth.

As can be judged from the list of abstracts that follow, there was also great variety and interest within the programme at large.

The Annual General Meeting held within the conference timetable confirmed that the venue for BCTCS6 would be Manchester University. BCTCS6 will take place 28 - 30 March 1990. Further particulars can be obtained by writing to Jane Gray Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL.

The conference was generously sponsored by the IBM United Kingdom Trust.

Alan Gibbons

BCTCS5

ABSTRACTS

PARALLEL ALGORITHMS ON WORDS

Alberto Apostolico

Purdue University - University of L'Aquila

Combinatorial algorithms on words such as string searching, string comparisons, detection of regularities in strings etc. form important primitives of computation. The design of such algorithms has been carried out mostly within established serial paradigms, such as RAM and Turing Machines. In computationally intensive applications (e.g. biomolecular sequence comparisons, text retrieval, stereo matching), serial computation is inadequate, so that efficient parallel algorithms need to be developed. Some such algorithms have been designed in recent years, and are reviewed in this talk. In general, these algorithms have been developed from scratch, since attempts at parallelizing efficiently the best known serial solutions usually seem to fail.

MAKING FORMALISM WORK FOR US

Roland C. Backhouse

University of Groningen

Formal reasoning is notoriously long and arduous; in order to use it to reason effectively in the construction of programs it is, therefore, paramount that we design our notations to be both clear and economical. Taking examples from imperative programming, from the use of the Bird-Meertens formalism and from category theory we demonstrate how the right choice of what to denote and how it is denoted can make significant improvements to formal calculations. Brief mention is also made of the connection between economical notation and properties of type.

A PROMISING APPROACH TO COMPLEXITY MEASURES FOR COMPUTATION OVER SPLAY TREE CONJECTURES – RECENT ADVANCES

Richard Cole

New York University and Ecole Normale Superieure

Sleator and Tarjan devised a self-adjusting binary search tree, called the Splay Tree, that is competitive with many of the balanced tree schemes for maintaining a dictionary. They conjectured that the Splay algorithm is optimal, in an asymptotic sense, among the class of self-adjusting algorithms that use only rotations to reorganize a binary search tree. A special case of this conjecture is the Dynamic Finger Conjecture; it asserts that the Splay algorithm is optimal, in an asymptotic sense, among the class of algorithms for performing finger searches.

Tarjan also conjectured that, when used to implement a deque, the Splay Tree uses constant amortized time per operation; this conjecture, called the Deque Conjecture, stemmed from the proof, by Tarjan, of the analogous result for stacks.

We report results concerning the Dynamic Finger Conjecture and the Deque Conjecture.

COMPUTATIONS IN GALOIS THEORY

John D. Dixon

Carleston University, Ottawa and Oxford University

The talk will be a survey (accessible to non-specialists) of approaches to some computational questions which arise in Galois theory and the study of algebraic number fields.

Suppose that $f(X)$ and $g(X)$ are polynomials of degree n with integer coefficients and roots α and β, respectively. Among the questions which we consider are to what extent we can solve the following problems efficiently:

(Q1) Factor $f(X)$ into its irreducible factors over $Z[X]$, and in particular decide whether $f(X)$ is irreducible (or equivalently whether $[Q(\alpha) : Q] = n$).

(Q2) In the case that both f and g are irreducible, decide whether $Q(\alpha)$ and $Q(\beta)$ are isomorphic fields.

(Q3) Compute the Galois group Gal(f) as a permutation group on the roots of f. (this group may have order as large as $n!$.)

(Q4) Find the subfields of $Q(\alpha)$ of degree d over Q for specified d.

OBJECT ORIENTED PROGRAMMING AS A NATURAL EXTENSION OF FUNCTIONAL PROGRAMMING

Joseph A. Goguen

Programming Research Group, Oxford

The simplicity and elegance of pure functional programming is marred by the awkwardness and inefficiency that arises from not having states; for some applications, such as database systems, the problem is acute, while for others it is unimportant. On the other hand, ordinary imperative programming is easily abused and often opaque. Object oriented programming corrects some major problems with imperative programming, but existing languages tend to be complex and ad hoc. This talk shows how to design an object oriented language that is just as clean and simple as a functional language, by starting from a semantics that naturally extends the semantics of (first order) functional programming. In fact, the mathematics of algebras with hidden sorts is not only a natural extension of initial algebra semantics, but also of classical automaton theory, and the resulting language FOOPS is a natural extension of our existing functional language OBJ3. Thus software modules are abstract machines, which generalize abstract data types. A more powerful reflective semantics is also given. Both semantics support multiple inheritance, generic modules, abstract data types, and the integration of specification and design with coding. Furthermore, graph rewriting techniques for implementing functional languages also extend naturally to the implementation of objects.

THE COPPO-DEZANI TYPE-SYSTEM

Roger Hindley

University of Swansea

The Coppo-Dezani System is a Type-theory for λ-calculus and functional programming. It is intermediate in expressive power between the simple but rather limited systems of Milner, Curry, Church, etc., and the strong polymorphic systems of Reynolds, Girard, Martin-Löf, etc.

Interest is at present focussed on strong systems, but I believe these will eventually turn out to be too space-demanding and slow to implement on smaller machines and there will be a need for systems of intermediate power. The Coppo-Dezani system is the start of an attempt to answer this need.

But besides a practical motivation, it has some very neat and interesting mathematical properties. These will be outlined in the talk.

CURRENT THEORETICAL ISSUES IN LOGIC PROGRAMMING

John Lloyd

University of Bristol

In this talk I will discuss various current issues of importance in the theory of logic programming, concentrating particularly on what I see as a mismatch between theory and practice. By this, I mean that much of the theory of logic programming only applies to pure subsets of Prolog, whereas the extra-logical facilities of the language appear essential for it to be practical. This mismatch is especially evident in existing theories of program transformation, partial evaluation, dubugging, and so on. I will discuss current research which attempts to reduce this mismatch.

THE THESIS THAT COMPUTABLE FUNCTIONS ARE UNIFORMLY CONTINUOUS

Michael B. Smyth

Imperial College

A promising approach to complexity measures for computation over infinite data – in particular, infinite precision real numbers – proposes that complexity be measured by a function m such that, to obtain k bits of output, an amount $m(k)$ of computation suffices. This implies that, to be properly computable - to have a complexity measure at all – a function must be uniformly continuous. We propose to adopt this principle, and to bring it into relation with Domain Theory. This means that domains have to be equipped with some kind of metric structure, in order to support notions of uniformity (and, thereby, of computational complexity). Having done this, we have to face the (drastic) effects of the requirement of <u>uniform</u> continuity.

UNIFYING DECLARATIVE AND PROCEDURAL PROGRAMMING PARADIGMS

Meurig Beynon
University of Warwick

In his celebrated Turing Award Lecture ("Can programming be liberated from the von Neumann style?"), John Backus discusses the need to complement the virtues of functional programming methods with means of achieving history-sensitivity. Despite developments in both declarative and procedural programming paradigms over the last decade, the problems raised by Backus remain significant. This talk considers the prospects for unifying declarative methods (as e.g. represented by pure logic and functional programming) and procedural methods (as e.g. represented by object-oriented programming) within a single programmi paradigm, based on the use of variable definitions that in their simplest form resemble the definition of values in a spreadsheet.

WELL FOUNDED INDUCTION IN A PROGRAMMING-BY-PROOF SYSTEM

Georges Werner and El-Hassan Bezzazi (speaker)
Laboratoire d'informatique fondamentale de Lille, France

A first order formal system for programming with proofs is given in this paper. In this system data types are defined as sorts of free algebras and the specification of the problem is stated with axioms and well founded relations. The induction schemes defined for these relations reflect the recursive part of the function to be defined. This is done with the realisability interpretation.

THE SOUNDNESS AND COMPLETENESS OF AXIOMS FOR CSP PROCESSES

Stephen Blamey
University of Oxford

I want to suggest that the failures model for CSP, and any modification or extension of that model, should be provided with a precise 'implementational' definition delineating which structures of the relevant logical type count as representations of processes—a definition which directly reflects an intuitive understanding of the modelling, and against which a set of explicit axioms should be required to be sound and complete.

The classic failures model (Brookes, Hoare and Rocsoe 1984) and the failures/divergence model (Brookes and Roscoe 1985) will both be given an implementational definition, and the standard axioms will be proved sound and complete. But the guts of the paper concerns the infinitary models that Roscoe (1988) has been motivated to consider in order to handle the infinite nondeterministic composition of processes in a satisfactory way. The apparatus of an implementational definition is very useful here to keep track of what's going on and to make comparisons with the original finitary models. Roscoe's proposed axiom set could actually be seen to import the implementational defintion in an indirect way, and its completeness is therefore rather trivial. In contrast, I shall propose axioms which are conceptually independent of an implementational definition, but which are (non-trivially) sound and complete with respect to it.

THE STRUCTURE OF COMMUNICATION COMPLEXITY

Andrew Chin
Mathematical Institute
University of Oxford

The advent of highly parallel computer architectures in the beginning of this decade raised questions about the absolute limitations of circuitry in the domain of very large scale integration (VLSI). The answers found to date leave enormous scope for further research. In the meantime, the most effective too in this study - the theory of communication complexity - has emerged in its own right as a fundamenta structural measure possessing a rich theory.

I report selected recent results connecting the communication complexity measure to familiar topics in combinatorics and computational complexity. This work supports my expectation that the communication complexity measure will, in time, shed considerable light on the value of parallelism.

INTENSIONAL EQUALITY + TYPE TAGS = A DERIVATION-LESS CONSTRUCTIVE TYPE THEORY

Mark M. Christian
Queen Mary College, University of London

Almost all research into Per Martin-Lof's Constructive Type Theory has centred in the so-call extensional theory. In such a theory judgments come with, indeed are the conclusions of, derivations. Judgments cannot be made in isolation since not all the information contained in a derivation is reflected in the final judgment.

We see this as deficiency for two reasons. Firstly derivations are a considerable burden to be negotiated on the way to producing programs that run on real machinery. Secondly when using pencil and paper one is tempted to ignore derivations and this leads to the possibility of writing incorrect judgments.

Our approach is to use to the intensional theory, equality at least is now decidable. However this is still not enough, Firstly elimination rules throw away the type they are eliminating over. Secondly evaluation of type can loose information as something general becomes particular. A possible solution is to put this lost information into the non-canonical forms. We get decidability but the size of term increases unacceptably. We present a version of Constructive Type Theory within which this extra information is put in only if it cannot be inferred from context.

ORDINAL THEORY AND TERMINATION

E.A. Cichon
Royal Holloway and Bedford New College
University of London

The purpose of this talk is to give a concise introduction to the theory of ordinal notation systems and some applications to Proof Theory and to Term Rewriting.

A notation system for the ordinals below the so-called ordinal GAMMA-0 is described together with some of the traditional uses of these (notations for) ordinals in Proof-Theory, in particular with respect to Subrecursive Hierarchies of Number Theoretic Functions and to Independence Results.

A selection is taken of the results presented in the paper PROOF-THEORETIC TECHNIQUES FOR TERM REWRITING THEORY (N.Dershowitz and M.Okada) which gives some of the connections between ordinals and ordering structures used in proofs of termination of term-rewrite systems.

TWO-WAY PATTERN MATCHING*

Maxime Crochemore

* joint work with Dominique Perrin

Université Paris 7, L.I.T.P., 2, Place Jussieu, F-75221 PARIS.

ABSTRACT: A new string-matching algorithm is presented, which can be viewed as an intermediate between the classical algorithms of Knuth, Morris and Pratt on the one hand and Boyer and Moore on the other hand. The algorithm is linear in time and uses constant space as the algorithm of Galil and Seiferas. It presents the advantage of being remarkably simple which consequently makes its analysis possible. The algorithm relies on previously known results in combinatorics on words.

A NEW VIEW OF BINARY TREES

Jeremy Gibbons
Programming Research Group
University of Oxford

We present a new formalism of labelled binary trees, using two partial binary constructors instead of the usual total ternary constructor. This formalism is complemented by some higher-order operators, encapsulating common patterns of computation on trees. We explore their use by deriving solutions to a couple of algorithmic problems on trees.

The work proceeds in the Bird-Meertens style. This is a calculus for programs, closely related to applicative programming languages. Expressions are written at the function level, avoiding the use of unnecessary names and being as concise as possible. Derivations are performed by program transformation — the application of correctness-preserving transformations to an inital specification, with the intention of improving its efficiency without changing its meaning.

A KRIPKE-LIKE MODEL FOR NEGATION IN LOGIC PROGRAMMING

James Harland
University of Edinburgh

We extend the Kripke-like model theory given by Miller for a fragment of first-order hereditary Harrop formulae (that is, the basis for first-order λProlog) to include negated atoms. This gives us a formal framework in which to study the addition of negated atoms to first-order hereditary Harrop formulae, and also the role of Negation as Failure rule. The class of predicates for which Negation As Failure is applicable is discussed, as well as the predicates for which some other form of negation will need to be used. We show how both classes may be incorporated into the model theory, giving a generalisation of the usual T^{∞} construction. No restriction on the class of programs is needed for this approach; the construction may be used for programs which are not locally stratified in the sense of Przymusinski. This is accomplished by the use of a *success level* and a *failure level* of a goal, either or both of which may be infinite. The resulting T operator is not monotonic, which necessitates a slight departure from the standard procedure, but the important properties of the construction still hold.

WHEN ASYNCHRONOUS COMMUNICATION IS EAGER EVALUATION

Christopher M. Holt
University of Newcastle-upon-Tyne

The development of mathematical models of communication has often focussed upon synchronized, handshake message passing, because a given transaction can then be understood within a single time frame. Recent efforts to develop models of asynchronous communication reflecting actual usage have found it difficult to apply much of the standard apparatus of mathematics due to the effects of introducing time. It is noted that for a number of communication dependency graphs, the two kinds of communication lead to identical behaviours, making asynchronous communication mathematically tractable. Graphs that do not have this property can be understood as inherently more complicated; this provides a means of measuring the complexity of a communications network. Since (except when dealing with alternative inputs) asynchronous graphs can be mapped into synchronous ones by adding response communications, it is suggested that where order is important such conversions be standard practice to ensure that communications networks are "well-structured". The problem of alternatives can be resolved by ensuring either that such an alternative is within a loop, or by demonstrating that the loss of a communication is immaterial to the program, corresponding to the case in which the sending process is forcibly terminated from the outside (killed off).

"The Expressive Power of Constraint Networks"

P.Jeavons, RHBNC

Networks of interlocking local constraints arise naturally in a number of diverse applications: machine vision, database retrieval, problem solving. Algorithms which calculate global solutions satisfying given constraints have been extensively studied. But just how powerful are constraint networks? We discuss an analogy with formal languages, and present some quantitative results about the expressive power of binary constraints. We also describe a natural implementation of such a network in a connectionist architecture for machine learning, and show that it has intriguing properties.

FAIRNESS VS. COMMUNICATING SEQUENTIAL PROCESSES

Alan Jeffrey
Programming Research Group
University of Oxford

Hoare's Communicating Sequential Processes (CSP) is a process algebra for describing concurrent systems. Current models take a very pessimistic view of livelock, identifying any process which might diverge with one that will. This results in many real-world algorithms (such as ethernet) being equivalent to $\perp$, the completely broken process. One way of fixing this, without involving explicit probabilities, is to introduce the notion of *fairness*.

In a model such as CSP where concurrency can produce deadlock, the usual notion of fair infinite behaviour breaks down. If we keep infinite traces as the only record of a process' infinite behaviour, we lose fairness. Another method of looking at fairness will be presented, based on the notion of strategies playing against processes. This allows a fair denotational semantics for CSP to be constructed.

FAST UNIFORM GENERATION OF GRAPHS WITH GIVEN DEGREE SEQUENCE

Mark Jerrum and Alistair Sinclair
University of Edinburgh

An algorithm is presented which randomly selects a labelled graph with specified vertex degrees from a distribution which is arbitrarily close to uniform. The algorithm runs in polynomial time for a wide class of degree sequences, including all regular sequences and all n-vertex sequences with no degree exceeding $\sqrt{n/2}$. An efficient approximation algorithm for counting such graphs is also presented. The method is based on simulation of a rapidly convergent stochastic process.

INFINITARY TRACE LANGUAGES AND FAIRNESS

Marta Kwiatkowska
University of Leicester

Fairness in a non-interleaving semantic model for concurrency has been investigated. In contrast to the interleaving approach, which reduces non-sequential behaviours to a non-deterministic choice between possible interleavings of activities of concurrent processes, concurrency and causality were assumed as primitive notions. Mazurkiewicz's trace languages were chosen as behavioural representations of systems and Shields' asynchronous transition systems as their acceptors. The notion central to these two formalisms is one of causal independency, which determines trace equivalence (congruence) in the monoid of strings. Equivalence classes of strings are called traces. The quotient monoid of traces forms a poset with trace prefix ordering.

First, trace languages have been enhanced to allow for infinite traces; this was achieved by introducing trace preorder relation on possibly infinite strings. It has been shown that the extension gives rise to the domain of traces and an infinitary monoid. Asynchronous transition systems have been equipped with a notion of a process structure. In this setting, a topological characterization of behavioural properties which includes safety, progress and fairness properties has been provided. Unconditional process fairness properties that are determined by process structures have been distinguished; they form a lattice with inclusion ordering. Finally, strength predicates were incorporated to allow for a variety of specific fairness properties such as weak and strong process fairness as well as equifairness and state fairness.

MAXIMUM MATCHING IN PLANAR GRAPHS

Elias Dahlhaus, Marek Karpinski
Department of Computer Science, Bonn University, 5300 Bonn 1, West Germany
Andrzej Lingas
Department of Computer Science, Linköping University, 581 83 Linköping, Sweden

Abstract: Let $G = (V, E)$ be an undirected graph. A *matching* $M \subseteq E$ is a set of edges no two of which have a common endpoint. A *maximum (cardinality) matching* is a matching that has the largest possible number of edges. It is an outstanding open question in the complexity theory whether the maximum matching problem or its decision version are in the class NC, i.e. whether they admit parallel algorithms running in poly-log time and using polynomial number of processors. The fastest known, deterministic parallel algorithm for maximum matching in bipartite graphs, due to Goldberg, Plotkin and Vaidya, runs in time $O(n^{\frac{2}{3}} log^3 n)$ using a polynomial number of processors. *A perfect matching* of G is a matching which for every vertex v of G includes an edge incident to v. For planar bipartite graphs, the problem of finding a perfect matching has been recently shown to be in NC. However, the problem of finding a maximum matching in planar bipartite graphs remains open. We present a parallel algorithm for finding a maximum matching in an arbitrary planar bipartite graph G. If the size of a maximum matching l of G is large our algorithm is faster and more processor efficient than that due to Goldberg, Plotkin and Vaidya. It runs in time $O((n/2 - l + \sqrt{n}) \log^7 n)$ on a CRCW PRAM with $O(n^{1.5}/\log^3 n)$ processors.

ORDERING INVARIANTS AND TERMINATION

Ursula Martin
Royal Holloway and Bedford New College, University of London

Termination of string rewriting systems (Semi-Thue systems) is proved by constructing particular kinds of well-founded orderings on strings. In this talk we present some of the orderings which have been studied. One such involves ordering strings by mapping them to the multisets consisting of their elements; these orderings are classified by cones. However they cannot order two strings whose multisets are the same. We outline a new construction which can always discriminate between strings, which relies on constructing cones in the factors of the lower central series of free groups. As a by-product we obtain invariants for rewriting systems which are proved terminating by the orderings.

ON MINIMAL SOBER SPACES

Aisling E. McCluskey
Queen's University, Belfast

Abstract (preferably typed, not more than half a page, on a separate sheet if prefered):

Those topologies which are minimal sober and minimal (sober and T_D) are characterised as being nested. In addition, those topologies which are minimal (sober and T_{ES}) and minimal (sober and T_{EF}) are identified.

SEMANTIC FIXED POINTS FOR SOME MODEL LOGICS

Nicholas Measor
University of Leicester

If it is desired to add the capacity for self-reference to some logical notation, a traditional approach is to axiomatize the logic in such a way that the existence of fixed points, explicitly definable in terms of the basic vocabulary, can be demonstrated proof-theoretically. A well-known example is the "Löb system" KW in modal logic.

An alternative approach is to introduce λ-abstraction and a primitive operator fix into the logical notation and then adjust the underlying semantics so that fix $(\lambda p.\varphi) \Leftrightarrow (\lambda p.\varphi)(\text{fix } (\lambda p.\varphi))$is valid for all φ. The talk compares the two strategies and explores the prospects for the latter, which, we suggest, is likely to prove more fruitful, at least in the context of epistemic logic.

ON THE FAIR SEMANTICS OF UNITY PROGRAMS

Dominique Mery
CHRS-CRIN, Universite de Nancy I, France

A UNITY [CM 88] program consists of a declaration part, a specification part of initial conditions, an always part and a finite set of actions or units. The basic assumptions of UNITY programs execution are the execution of only one action at any time and the fair execution of UNITY programs. The goal of UNITY programming style is to give a foundation of concurrency and related notions to obtain solutions on specific architectures. A crucial hypothesis is the fairness hypothesis: it means that any action is eventually activated. Our previous work [MER 87]. [MP 86] have semantically characterized eventually properties under fairness hypothesis and have allowed to build a sound and semantically complete proof system. The main result was the construction of the greatest assertion leading to a given assertion under fairness hypothesis. We characterize the UNITY fairness by modifying our previous operator. The computation of the set of states eventually leading to the termination of UNITY programs is based on the double and successive computation of least and greatest fixed points as PARK proposed in [PAR 81]. We identify special units that help to obtain termination. Finally, a very simple example illustrates our work.

Our communication associates some kind of weakest precondition to a terminating UNITY program and we define a mathematical framework to express the fair semantics of UNITY programs. Our study is very useful to characterize criteria for a semantically complete proof system for UNITY programs.

TAXONOMIES AND CHALLENGES IN CONCURRENCY THEORY

David V.J. Murphy
University of Surrey

We believe that a full understanding of the varieties of concurrency theory can only be achieved by appreciating how they are related. The first half of this paper presents a taxonomy of models of what are called concurrent systems, – we prefer to discuss *abstract, reactive* systems. Theories are classified by their view of time (linear, branching or partial order), by the degree of introspection that is allowed into the mechanism of processes, and by the nature of events or transitions in the model. The aim of this taxonomy is to provide a framework for choosing a model based on notion of behaviour that model must respect. We apply this classification to a number of theories of abstract, reactive systems.

The second half of the paper examines a new theory of concurrency, *interval event structures*. This model was conceived as a descriptive and semantic tool in the study of detailed real-timed behaviours. We show how the main features of the model were formed in response to a specific notion of behaviour and of behavioural equivalence. We also discuss the insights this model gives into some standard preoccupations of concurrency theory.

LOGICS OF ANNOTATIONS FOR CONCURRENCY

Pawel Paczkowski
University of Edinburgh

Annotations are defined as finite labelled transition systems with predicates of first order logic as configurations and atomic actions of a while-programming language as transition labels. Annotations are used for proving partial correctness, termination and deadlock freedom.

Completeness results are obtained under usual expressiveness assumptions despite that in contrast to extensions of Hoare Logics for concurrency no use is made of auxiliary machinery (auxiliary, history or location variables, environment assumptions).

Annotation based proof methodology which can be viewed as a generalization of Floyd's program verification technique is proposed as an object for further generalizations and development in order to modularize proofs.

PLANAR ACYCLIC COMPUTATION

Mike Paterson
University of Warwick

Restricting acyclic Boolean circuits to two dimensions is a severe limitatior
Although for most bases we may design 'crossovers', i.e., planar subcircuits simulating the crossing of a pair of wires, such a simulation is deficient ir one important respect.
The computation of some functions may become more expensive using planar circuits, while for some sets of functions with input and output locations specified the computation may become impossible.
We characterize those input/output specifications which are realizable with planar circuits.

(This is joint work with Bill McColl of Oxford and Brian Bowditch of Warwick.

Adding a stability operator to CCS

Iain Phillips
Imperial College

Milner has taken the view that two processes should be deemed equivalent if they cannot be told apart by any observer of their external behaviour. De Nicola & Hennessy showed how to realise this idea for Milner's language CCS of communicating processes by taking the observers to be CCS processes, thus allowing the language to test itself. There are circumstances in which it is reasonable to regard the refusal of a process to perform an action as (finitely) observable. A corresponding formal language of refusal tests can be given, giving rise to so-called refusal equivalence on CCS processes. We propose adding a new operator to CCS which has the effect of observing whether a system is stable, that is, incapable of further spontaneous action, and show how refusal tests can be interpreted in the augmented language, allowing refusal equivalence to be recovered by letting the new language test itself in De Nicola-Hennessy style.

INDUCTION RULES FOR NON-INDUCTIVE TYPES IN TYPE THEORY

Ruy J.G.B. de Queiroz and Michael B. Smyth
Imperial College

In previous works (de Queiroz 198?bc) it has been argued that the meaning of a type in a type-theoretic framework is made explicit by the corresponding *reduction* rules. This leaves open the question of how to explain the rôle of certain equality rules which are essentially different from *normalisation* (*reduction*) equality. The η-rule of the λ-calculus, e.g., is an equality rule where the nature of equality appears to be of a very different character from β-rule equality. Let us call these, for the moment, 'special' equality rules for non-inductive types such as Π and Σ. The semantical framework which provides the basis for the argument concerning the rôle of the *reduction* rules is a particular type-theoretic based account of meaning for the mathematical language named 'semantics of use', *use* being represented by rules of *normalisation* (de Queiroz 198?a, 1988b). When attempting to say that the meaning of Π-types, whose elements are λ-terms, can be explained by the β-*normalisation* rule alone, one is almost inevitably confronted with the question of what rôle the analogue of λ-calculus η-rule is meant to play in a type-theoretic framework. In this paper we suggest that the right way to view the 'special' equality rules is as formalised induction rules for non-inductive types. It will be shown that their rôle is, in effect, to ensure *minimality* of the type. The analogue of η-rule for Π-types seems to be useful to formalise the usually informal *inductive* clause of the definition of types by saying that 'every element of the type is a λ-*abstraction* term', and therefore, no meaning beyond what is indicated by the β-rule is to be taken as the meaning of Π-types.

ON THE COMPLEXITY OF APPROXIMATION LANGUAGES

Jose D.P. Rolim
Odense University, Denmark

The drawbacks of the worst-case complexity have stimulated several attempts at developing a more unified complexity theory. These attempts have been based mainly on *almost everywhere* conditions. In this paper we consider bounds that do neither necessarily hold almost everywhere nor are necessarily worst-case bounds. We consider functions that are in between the worst and the best-case bounds and bounds that hold *infinitely often.* The approach to complexity theory herein defined is named *IO-complexity*, with *IO* standing for infinitely often. We show the connection between these notions and the problem of *approximately* solving a problem.

We assume a probability distribution $P[X = w/n]$ on inputs of length n and we introduce the concept of *density functions* to compare bounds. Thus, for example, we say that a Turing machine M is of *IO*-time complexity $f(n)$ with density function $d(n)$ and probability distribution $P[X = w/n]$ if $d(n)$ is a lower bound for the sum of the probabilities of all words of length n for which M halts in time $f(n)$. We identify the density function $d(n) = 1$ case with the worst-case complexity and we show that most results of the worst-case complexity can be translated into analogous results for the *IO*-complexity with any density function $d(n)$.

We give formal definitions for the notion of finding approximation solutions for hard problems. We consider *approximation languages* that are required to 'solve' part of the hard problem, i.e. we require finding the correct answer only for some inputs. We show an interpretation of the *IO*-complexity theory in terms of approximation solutions. We show that the recursive languages of an *IO*- complexity class which resource bound $f(n)$ and density $d(n)$ are precisely those languages L which can be *approximated* by $f(n)$ bounded machines agreeing with L on input w with probability at least $d(|w|)$. These results when stated in terms of hierarchy of languages strengthen the classical hierarchy results; for example, we show the existence of problems 'easily' solved in bound $g(n)$ that do not admit *any* approximation solution in bound $f(n)$, with f and g such that $g(n) \geq cf(n) logn$ a. e. for any constant c.

We conclude by making conjectures and showing some preliminary results on properties that would tie the hierarchy problems and the open trade-off problems of standard complexity theory to those of approximation theory.

CATEGORICAL STRUCTURES FOR PROGRAMMING

B. Hilken and D. Rydeheard (speaker)
University of Manchester

We propose an algebraic structure containing the following constituents of programming: Programs themselves (treated as λ-terms), specifications, proofs of correctness, and conversions between programs e.g. program evaluation. The aim is to produce programming environments to support various techniques of program development. The structures we propose have a genericity, for instance of the underlying logic, different from that of LF and Isabelle, and based upon a common form for inference rules.

The basic concept is that of a *hyperdoctrine* introduced by Lawvere. Hyperdoctrines have recently been used to model 2nd-order λ-calculus. The suggestion here is to build upon the original application - describing logics in terms of categories, functors and universality. We extend this to 2-categories to include conversions between programs as suggested by Seely.

We then show how these structures support:

- Specifications of partial functions using pre- and post-conditions (in the VDM style) interpreted as a form of ϵ-quantification,
- Invariants and subtypes, equivalence relations and quotient types, recursion,
- Data abstraction via 'logical relations',

This is a preliminary announcement of work in progress.

THE COMPLEXITY OF TOTAL COLOURINGS

Abdon Sanchez
Linacre College, University of Oxford

SPECIFICATION AND VERIFICATION OF STATE MACHINES

Victoria Stavridou
Royal Holloway and Bedford New College, University of London

Formal methods are applied with increased frequency in the specification and verification of digital systems as an alternative to traditional methods of establishing correctness, such as simulation and testing. Our objective here is to report on the results of a controlled experiment comparing formalisms and systems that are currently used for formally specifying and verifying synchronous sequential circuits expressed as state machines. This work follows on from a previous case study on combinational devices and includes experiments with temporal logics, finite state machine languages, the Boyer-Moore theorem prover, the HOL system and the algebraic specification language OBJ3. The example used throughout is a twisted ring counter.

AN EXAMINATION OF GIRARD'S QUANTITATIVE DOMAINS

Paul Taylor
Imperial College, University of London

Girard showed that stable functors Set^A -> Set^B can be expressed as power-series, but did not characterise the function-space with the stable order. We carry through this programme, finding that we must use groupoid actions and not sets. A new model of Linear Logic is found (including of-course) and this category of domains is itself a domain (type-of-types).

GROUPS AND FORMAL LANGUAGES

Rick Thomas
University of Leicester

A finitely generated group G may be specified by a *presentation* $< X : R >$ in which X is a finite set of *generators* and R is a set of *relations*. If we let $X^{-1} = \{x^{-1} : x \in X\}$ and $\Sigma = X \cup X^{-1}$, then we may define an equivalence relation $\equiv$ on Σ^* by $\alpha \equiv \beta$ if and only if α and β represent the same element of G; every element in G thus corresponds to an equivalence class of $\equiv$, and every subgroup S of G to a union of such classes. Moreover, if S is a subgroup of G, we may generalize this idea and consider subsets Ω of Σ^* such that every element of S corresponds to at least one element of Ω and every element of Ω represents an element of S.

Formal languages are often classified in terms of their complexity; in particular, we have the *Chomsky hierarchy* of languages. Given the above discussion, if we have a subgroup S of G and a corresponding subset Ω of Σ^*, a natural question to ask is whether the fact that Ω forms a particular type of language corresponds to a natural group-theoretic property for S. The aim of this talk will be to introduce these concepts, explain the terms used and mention some of the results already achieved in this area.

SPECIFICATION AND COMPUTATION IN LOGIC PROGRAMMING

J.V. Tucker
University of Leeds

The talk considers the computability of relations over classes of many sorted algebras. The aim is to establish a Church-Turing Thesis for specifications for computation on abstract data types. In addition to logic programming, other models will be considered, including while-array programs with initialisation, and random assignments. The results were obtained in joint work with J. I. Zucker (SUNY, Buffalo).

PROOFS AND PROGRAMS

S.S. Wainer
University of Leeds

A simple survey talk on basic ideas from Mathematical Logic relating to the connection between proof-rules and program-constructs. Also some applications measuring (in proof-theoretic terms) the complexity of some standard program-transformations.

COMPOSITIONALITY THROUGH AN OPERATIONAL SEMANTICS OF CONTEXTS

Kim G. Larsen and Liu Xinxin
Aalborg University, Denmark

In this paper we intend to provide a theoretical foundation for top-down design methodology for reactive systems. The problem under consideration is the following:

> Given a specification in which the desired property of the final program is described. Also, given a context which is a (unfinished) program with some unknown parts left to be designed. How to derive the properties that these unknown parts should satisfy, in order that the final program satisfy the overall specification.

We would like the derived properties to be as weak as possible, in order not to limit the choice for further implementation. We also want these properties to be decomposeable such that they can be expressed as seperate properties, each of which should be satisfied by one of the unknown parts. To deal with the problem, a new operational semantics of contexts in forms of action transducers is given. A version of Hennessy-Milner Logic extended with recursion is used for the specification language. In this setting, we solve the above problem.

ON SOUNDNESS AND COMPLETENESS OF CALCULI FOR EQUATIONS, DEPENDENT EQUATIONS AND QUASI-DEPENDENT EQUATIONS

Sun Yong
University of Edinburgh

This talk concentrates on calculi for equations, dependent equations (or equational implications, conditional equations) and quasi-dependent equations (or quasi-equations). They are written as $\tilde{D}$, $\tilde{D}^d$ and $\tilde{D}^q$ respectively.

As a summary, we have sound and complete deduction systems $\tilde{D}$, $\tilde{D}^d_-$ and $\tilde{D}^q_-$ for Σ-equations, dependent Σ-equations and quasi-dependent Σ-equations with respect to Σ-equations.

These results are good enough for us to have natural deduction systems corresponding to them. The reason for this comes from the fact that $\sqcup\tilde{D}(\tilde{\Gamma}) = \bigcup_{n \in Nat} \tilde{D}^n(\tilde{\Gamma})$ for every $\tilde{\Gamma}$, $\sqcup\tilde{D}^d_-(\tilde{\Gamma}) = \bigcup_{n \in Nat} \tilde{D}^{d^n}_-(\tilde{\Gamma}^d)$ for every $\tilde{\Gamma}^d$ and $\sqcup\tilde{D}^q_-(\tilde{\Gamma}) = \bigcup_{n \in Nat} \tilde{D}^{q^n}_-(\tilde{\Gamma})$ for every $\tilde{\Gamma}^q$.

About $\tilde{D}^q$, I discover that Birkhoff's approach is not powerful enough to totally capture it. Therefore, to conclude this talk, I name the existence of $\tilde{D}^q$ as an open problem and raise a closely-related question below.

Question 1 : Whether is there a condition, under which $\tilde{D}^q_+$ totally captures valid quasi-dependent Σ-equations?

Open Problem 2 : Is there a sound and complete deduction system $\tilde{D}^q$ for quasi-dependent Σ-equations?

A TECHNIQUE FOR EFFICIENT IMPLEMENTATION OF CERTAIN RECURSIVE P-RAM ALGORITHMS ON MESH-CONNECTED COMPUTERS

Ridha Ziani (with Alan Gibbons)
University of Warwick

Let P be a recursively described and distributable problem of size n. We describe a general strategic approach which will often make an $O(\sqrt{n})$ parallel time solution possible on a 2-dimensional mesh-connected computer with O(n) processors. Such a solution is time-optimal. It is likely that the class of such problems is large and will contain many that are non-trivial. We illustrate the technique with reference to a number of problems. The methodology is easily extended to mesh-connected computers of arbitrary dimension to again obtain time-optimal solutions using O(n) processors.

REPORT ON THE 2. GDR-COLLOQUIUM ON COMPUTATIONAL THEORY (BERLIN. 4.5.1989)

The second national one-day workshop on computational theory took place in Berlin, on 4.5.1989. As its predecessor in 1988 it offered a representative impression on what is being done in the last year in theoretical computer science in GDR. The papers presented on the workshop where rather successful. Some of them have been already