THE UNIVERSITY OF
WARWICK

Original citation:
Paterson, Michael S. (1993) Computer science seminars 1992/93. University of Warwick. Department of Computer Science. (Department of Computer Science Research Report). (Unpublished) CS-RR-249

Permanent WRAP url:
http://wrap.warwick.ac.uk/60931

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

A note on versions:
The version presented in WRAP is the published version or, version of record, and may be cited as it appears here.For more information, please contact the WRAP Team at: publications@warwick.ac.uk

warwickpublicationswrap
highlight your research

http://wrap.warwick.ac.uk/

Computer Science Seminars 1992/93

compiled by

M.S. Paterson

Department of Computer Science
University of Warwick

Introduction

This report contains the abstracts of the programme of weekly seminars in the Department of Computer Science for the 1992/93 academic year. There were 28 seminars in all: 17 by external speakers and 11 from within the Department. With few exceptions, the seminars were held in the Department's Seminar Room on Tuesday afternoons at 4pm after an informal gathering for refreshments.

Programme and Abstracts of Talks

13 October, 1992

A Parallel Graph Reduction Machine

Mike Joy

Department of Computer Science

A description of the parallel graph reduction machine, and the associated functional language (Ginger), in use at Warwick is presented. The machine is a simulated shared-memory processor, and Ginger includes parallel constructs which are applicable both to shared-memory and distributed-memory computational models. Results obtained with the current system are described, together with the work currently under way to implement the machine on the Department's Parsytec parallel processor.

20 October

Victorian Data Processing:
the Census, the Clearing House, and the Prudential

Martin Campbell-Kelly

Department of Computer Science

Large-scale data processing started in England long before the commercial availability of office machines in the 1890s. This seminar describes the emergence of centralized data processing at three typical bureaucracies in the middle of the nineteenth century - the Census (c.1840), the Railway Clearing House (c.1850), and the Prudential Assurance Company (c.1860). All of these organizations employed several hundred clerks for data-processing tasks that would today be be performed by computers. The organizational and information-processing innovations are described, and related to twentieth century developments. The role of data processing in the wider Victorian economy is also discussed.

27 October

Self-Timed Circuits

Neil Wiseman

Computer Laboratory
University of Cambridge

There has long been the hope that asynchronous logic would allow an existing circuit technology to achieve the fastest operating speeds. Over some 35 years various attempts to establish a methodology for asynchronous logic have found few practitioners and the practical results have been disappointing. But at last something useful seems to be happening and the talk will present some of the recent ideas, as well as some older ones, that together may lead to large and fast systems becoming easier to design and assemble.

3 November

Software Development using Definitive Scripts:
Experiments and Observations

Meurig Beynon and Simon Yung

Department of Computer Science

A definitive script is a set of definitions that describes the dependencies between the values of procedural variables. In typical use, a script represents relationships between observations of a physical object whose state can be transformed by experiment.

This talk will describe how the study of definitive scripts has led us to look at software development in a new light. The main themes of the talk, to be illustrated by examples of software developed using our approach, are:

- foundations for programming in observation and experiment
- programming as modelling
- what is a program?
- design vs simulation
- from agents and privileges to protocols
- synchronous propagation of state-change
- new abstractions for state.

The software experiments on which our observations are based include experiments in:

- design and modelling of objects
- concurrent systems simulation
- reactive systems specification
- translating definitive models into procedural programs
- abstract development of functional programs

The aim of the talk is to assess the prospects for future development of our concepts and techniques as a new basis for software construction.

10 November

Autostereo Display

Neil Dodgson

Computer Laboratory
University of Cambridge

A new autostereoscopic 3D display has been developed as a joint project of the Computer Laboratory and the Engineering Department at the University of Cambridge. The device produces a three-dimensional effect without the need for special glasses or other aids, and allows the viewer(s) to look around objects simply by moving their head(s). This seminar will discuss the principle of this 3D display, how it is actually implemented, and the uses to which it is being put, including 3D interactive computer graphics.

12 November (Thursday)

Parallelism in 3D Graphics

Mikki Larcombe

Department of Computer Science

In the overall pattern of 3D graphics there are 3 phases; firstly we have excerption, deciding which parts of the world model will in fact be visible; secondly consecution, deciding in which order to render or paint the items extracted by excerption to give the correct appearance of depth; and finally the rendering itself where the surface primitives of the world model are translated into coloured and shaded pixels. The use of recursive convex space division structures provides a powerful tool for both excerption and consecution but it is not immediately apparent how this approach can be efficiently executed on a parallel machine. Rendering on the other hand has been considered an obvious candidate for parallel approaches but realities of hardware and of database access contention have bedevilled simplistic approaches (such as ray-tracing on a processor per pixel model). The seminar will describe an approach for "painters algorithm" visualisation which powerfully exploits parallelism by subdividing the view volume, not just in terms of subdividing the image plane but also in depth, allowing a proportional speedup, particularly in consecution. The depth division process avoids many of the problems of database access contention by providing the parallel processes with their appropriate parts of the database, rather than processes seeking their relevant data.

17 November

Efficient Program Transformations for Responsive Parallel Computing

Krishna V. Palem

IBM T. J. Watson Research Center
Yorktown Heights

High-level parallel programming languages are designed to present a simple virtual machine model to the programmer. This allows the programmer to focus on the essentials of the correctness and efficiency of programs, and thereby improves productivity. However, engineering constraints do not always allow these idealized abstractions to be directly implemented in practice. A natural approach to bridging this gap is via mechanisms for automatically transforming the parallel algorithms or programs designed in the (idealized) high-level language to execute correctly on machines that can be actually realized in practice.

In this talk, we will describe program transformations which provide a basis for the compilation of (ideal) *synchronous* programs to run efficiently on *asynchronous* parallel machines with shared memory. The PRAM model for algorithm design, and the virtual machine of the UC programming language, are instances of the ideal programming model considered here. In addition to being correct and efficient, the transformed programs have the important property that they are *responsive.* Essentially, responsive computations always make progress on the asynchronous target machine without waiting for slow or failed processors to complete their work. Therefore, responsive executions are *fault-tolerant* in the following strong sense: the computation will terminate correctly even if processors fail arbitrarily often during the computation, so long as at least one processor is executing correctly at any time. No implicit assumptions about the relative speeds of processors, or about synchronization primitives built into the target machines, are required for our methods to work. By using randomization, we can guarantee that the executions on the asynchronous targets are guaranteed to require only $O(log n)$ extra space and

$O(log^3n)$ additional expected work, compared to the given n-way parallel source programs.

(This is joint work with Z. Kedem, M. Rabin and A. Raghunathan.)

24 November

Highly Efficient Asynchronous Execution of Large-grained Parallel Programs

Michael O. Rabin

Harvard University, Hebrew University
Visiting Fellow, Merton College and Oxford University

An n-thread parallel program $\mathcal{P}$ is *large-grained* if the program variables are "large objects" of size $\geq \log^3 n$, and/or its individual thread instructions in each parallel step are complex functions, requiring more than $\log^3 n$ machine operations. It is a theoretically challenging and practically important problem to ensure correct execution of $\mathcal{P}$ on an n-processor *asynchronous* parallel system. Let $\mathcal{P}$ be a large-grained program requiring total work W for its execution on a synchronous n-processor parallel system. In this talk, we present a transformation (compilation) of $\mathcal{P}$ into a program $T(\mathcal{P})$ which correctly simulates $\mathcal{P}$ on an asynchronous n processor system, requiring just $O(W \log^* n)$ total work. In the case that the program variables are large objects, the multiplicative memory (space) overhead is < 2. The solution involves a number of novel concepts and methods. These include the concept of a *uniform schedule* for an n-processor asynchronous machine, and the concept of a processor's *responsibility set* of computation threads. The latter concept, and its mode of application, opens the way for organizing the program data in memory so as to allow for computational locality considerations. The execution of $T(\mathcal{P})$ is *continually progressive*, which ensures strong fault-tolerance. The implications of the present work for data-processing programs are obvious. Our algorithm are randomized, producing both Monte Carlo and Las Vegas style results.

(This is joint work with Y. Aumann, Z. Kedem and K. Palem.)

1 December

Phase-space representations in signal analysis

Roland Wilson

Department of Computer Science

A phase-space signal representation is one in which one or more analysis co-ordinates are conjoined to the original co-ordinate system of the signal. For example, an audio signal, which is a function of time, may be represented using a Short-Time Fourier Transform (STFT), in which a frequency co-ordinate is conjoined to the original time co-ordinate, so that the signal is described as a function of the time and frequency co-ordinates. In recent years, several new types of phase-space representation, such as scale-space and wavelet transforms, have found use in signal analysis. Each of these representations can be computed by linear transformation of the signal; each approach has its advocates. So what are the significant differences between them and what do they have in common ? Why should we want to use such representations in the first place? Is there anything better? The aim of the talk is to shed some light on these issues, with illustrations taken from the work done by the image and signal processing group.

3 December

Large General Purpose Parallel Computing

Bill McColl

Programming Research Group, Oxford University

A major challenge for computer science in the 1990s is to determine the extent to which general purpose parallel computing can be achieved. The goal is to deliver both scalable parallel performance and architecture independent parallel software. (Work in the 1980s having shown that either of these alone can be achieved.) Success in this endeavour would permit the long overdue separation of software considerations in parallel computing, from those of hardware. This separation would, in turn, encourage the growth of a large and diverse parallel software industry, and provide a focus for future hardware developments.

In recent years a number of new routing and memory management techniques have been developed which permit the efficient implementation of a single shared address space on distributed memory architectures. We also now have a large set of efficient, practical shared memory parallel algorithms for important problems. In this talk I will introduce and discuss some of the current issues involved in the development of parallel computing systems which support fine grain concurrency in a single shared address space. The talk will cover algorithmic, architectural, technological, and programming issues.

8 January, 1993

The Extended Duration Calculus

Zhou Chaochen

International Institute of Software Technology
Macau

The talk will be about the extension of the Duration Calculus to deal with continuous systems, the so-called ‘hybrid’ systems that are of great interest now.

12 January

Specification and Verification of Fault Tolerant Designs

Jens Nordahl

Department of Computer Science
Technical University of Denmark

This talk presents an approach to specification and verification of fault tolerant designs based on CSP (*) and using CSP's trace logic specifications and its proof rules, rather than its processes and algebraic laws. The approach allows a designer of a fault tolerant system to specify assumptions on failure prone components, including global failure assumptions, i.e. assumptions on how many, and which components that are allowed to fail simultaneously. Based on these assumptions it offers a strategy for verifying that the design tolerates the specified faults.

Local assumptions on proper and failing behaviour are for each component given by the designer as two or more separate specifications (one specification for each considered behaviour type). Global fault assumptions can be specified either as constraints on parameters of local specifications, or as constraints on the occurrence of certain events. In either case, verification of system properties is carried out using the compositional proof rules of CSP.

21 January

Avoidance of Work-related Upper Limb Disorders

Peter Rayner

Univ. of Warwick Safety Officer

A brief explanation and description of the symptoms leading to and the methods of avoiding such injuries.

26 January

A brief history of Boolean function complexity

Mike Paterson

Department of Computer Science

I will explain some of the motivation for studying the complexity of Boolean functions and survey some old and new results in this area.

2 February

The asymptotic complexity of merging networks

Jun Tarui

Department of Computer Science

Let $M(m,n)$ be the minimum number of comparators in a comparator network that merges two ordered chains of lengths m and n where $n \geq m$. Batcher's odd-even merge yields the following upper bound.

$$M(m,n) < 1/2.(m+n)log(m+1) + O(n), \text{ e.g., } M(n,n) < nlogn + O(n).$$

We prove a new lower bound that matches the upper bound asymptotically:

$$M(m,n) > 1/2.(m+n)log(m+1) - O(m), \text{ e.g., } M(n,n) > nlogn - O(n).$$

Our proof technique extends to give similarly tight lower bounds for the size of monotone Boolean circuits for merging, and for the size of switching networks capable of realizing the set of permutations that arise from merging.

(This is joint work with Peter Bro Miltersen and Mike Paterson.)

9 February

Constraining Interference in a Design Method for Object-Based Concurrency

Cliff Jones

Department of Computer Science
University of Manchester

A development method (whether formal or otherwise) which is 'compositional' can increase productivity; 'interference' is what makes it difficult to find compositional development methods for concurrent systems. Earlier research on documenting rely and guarantee conditions did something to tame interference but the approach still required too much work to expect it to be used even as widely as methods like VDM. My recent research has used concepts from object-oriented languages to help constrain and structure the reasoning about concurrent programs. This seminar will concentrate on the idea of constraining the interference in a transformational development style and the delicate question of how to justify the transformation rules themselves.

16 February

Taming Infinite State Spaces

Colin Stirling

Department of Computer Science
University of Edinburgh

Process calculi such as CCS provide an elegant framework for modelling interactive systems. Complex systems can be described within such a calculus as an expression whose behavioural meaning is precisely determined by the rules for transitions, and two expressions may be deemed, for all practical purposes, to have the same behaviour when they are bisimulation equivalent.

In this talk I will examine some of the ramifications of bisimulation equivalence on process expressions which describe infinite state systems. In particular I will address the question: for which classes of systems is bisimulation equivalence decidable and why? Positive answers will be given for context free processes and a family of recursive parallel processes. These results depend on decomposition and cancellation properties. The intended ultimate goal of this research is find a process calculus with Turing power (the ability to generate all recursively enumerable languages) but for which bisimulation equivalence is decidable.

23 February

Intervals and Actions in a Timed Process Algebra

David Murphy

Department of Computer Science
University of Birmingham

This paper presents a timed process algebra, interval process algebra or IPA, with two novel features; nonatomic actions, and eager or urgent actions. Nonatomicity is achieved by associating actions with intervals of time. Timing allows us to distinguish between eager and lazy actions; eager actions are ones that happen at once, whereas lazy actions must wait until they are triggered. Having both sorts of action allows us to model systems with broadcast concurrency, and allows liveness properties to be accurately specified.

A novel timed operational semantics is presented for the calculus, and branching bisimulation is defined over it. We then show that recursive process specifications have unique solutions. Moreover our notion of equivalence is shown to be compositional and a congruence of action refinement.

We discuss an operation of closure that corresponds to putting a process in an empty environment. This allows us to associate a finite timed process graph with a finite process, and to give a form of expansion theorem for the calculus. The use of ill-timed but well-caused traces is crucial here; we discuss timing and causality in the calculus, and show that, under mild restrictions, all timed process graphs are the meanings of some process.

2 March

New Directions in Parallel Computing

Ian Watson

Department of Computer Science
University of Manchester

Truly general purpose parallel computing is still not a reality. Although there have been many different parallel computer structures and parallel programming approaches advocated over the last twenty or so years, the styles of problem which can be tackled sensibly in parallel are strictly limited.

This seminar attempts to identify the reasons for this and examines those directions which appear to offer hope for the future. The issues are both to do with machine architecture and programming languages although it is believed that the latter area, although of greater importance, has received less serious attention than the construction of high performance hardware.

New directions in programming language development being pursued at Manchester will also be outlined.

9 March

Using Formal Methods for Real Software

Anthony Hall

Praxis Systems plc

Software Engineering is the application of scientific principles to the economical development of software. It is, as yet, a poorly understood discipline and most software development is in fact still at the craft stage. One of the proposed remedies for this state of affairs is the use of Formal Methods, which use mathematics to express the properties of the software under development. These methods are highly controversial, and many people doubt the practicality of their application to 'real' projects.

Praxis is a software engineering company which does use formal methods, among others, for its development projects. On the basis of this experience we can understand what the real, as opposed to the theoretical, strengths and weaknesses of formal methods actually are. We are finding answers to questions like: Do formal methods work? If so, why? If not, why not, and what can we do about it? When and why are they better than more conventional methods? What sort of projects benefit from formal methods? What are the advantages and disadvantages of the different formal methods which have been proposed?

This talk will give an account of how formal methods are used in practice, and try to explain why they work and what their place is among the various software engineering techniques. It will explain in what sense formal methods are more scientific than others, and what this implies for the practising software developer.

16 March

Generalising the Cyclesum Test

Steve Matthews

Department of Computer Science

In 1981 Wadge introduced the "Cyclesum Test" for proving that certain Kahn Dataflow networks would not deadlock. Both the formulation of the Test and its proof were expressed in terms of Kahn's denotational semantic domain of finite and infinite sequences partially ordered by initial segments. In particular the work relied pivotally upon the notion of the "length" of a sequence. Unfortunately for the Cyclesum Test, in order to incorporate lazy evaluation, Lucid and its descendants use much larger domains than Kahn's. In such countable product domains finite sequences no longer exist, and so the Test cannot be formulated using the notion of length. A generalisation of length for more sophisticated domains is clearly needed to extend the Test to members of the Lucid family.

Following Wadge's suggestion that it might be possible to measure the "agreement" between objects in a domain this talk proposes "size" as the required generalisation of "length". The Cyclesum Test is reformulated using size and a proof given. Finally it is demonstrated how this process of generalisation has shed new light on aspects of both computability and the denotational semantics of functional languages.

27 April

A Practical Minimum Distance Method for Syntax Error Handling

Julia Dain

Department of Computer Science

> It is one thing to show a man that he is in an error, and another to put him in possession of truth.
>
> *Locke*: Essay on the Human Understanding

> Syntax error at or near line 1 - bailing out.
>
> *Aho, Kernighan and Weinberger*: awk

> Sorry, too many errors.
>
> *Stroustrup*: cfront

This talk presents a method for recovering from syntax errors encountered during parsing. The method provides a form of minimum distance repair, has linear time complexity, and is completely automatic. It is incorporated into the LR parser-generator yacc in such a way that a compiler writer can generate a parser with error recovery without providing any additional information to yacc. Error messages phrased in terms of source input are generated automatically.

A formal method is presented for evaluating the performance of error recovery methods, based on global minimum-distance error correction. The minimum distance method achieves a theoretically best performance on 80% of Pascal programs in the Ripley-Druseikis collection. Comparisons of performance with other syntax error recovery methods are given.

4 May

An Introduction to Descriptive Complexity Theory

Iain Stewart

Department of Computer Science
University College of Swansea

This talk is an introduction to some relatively recent novel applications of logic in complexity theory known as descriptive complexity theory (or finite model theory). Though this approach to complexity theory originated in a 1974 paper of Ron Fagin, it lay dormant until the early 80's, when it was taken up by Neil Immerman, amongst others. Its most notable achievement so far is probably that it was by using descriptive complexity theory that Immerman solved the longstanding open problem of whether the class of context sensitive languages is closed under complementation.

I shall outline the basic framework of the subject and provide a selection from the new results obtained so far. The talk should be especially accessible to non-specialists.

11 May

A State-Dependency Approach to System Design

Barry Dwyer

Department of Computer Science
University of Adelaide

The seminar considers a formal method of decomposing a system into a network of component systems connected by data flows. The decomposition often exposes opportunities for sequential or parallel access to data structures. The specifications of the system components can be derived from the system specification by a simple rewriting rule.

Most of the information needed to design a system can be visualised by means of an Attribute Dependency Graph, which is based on dependencies between state variables. The Attribute Dependency Graph for a system provides the designer with useful feedback, often leading to respecification of the problem.

18 May

Analysing coherence of intention in natural language dialogue

Paul McKevitt

Department of Computer Science
University of Sheffield

One of the problems in natural language processing is to build theories, models and implementations of how individual utterances cling together into a coherent and rational discourse. Current theories and models of natural language dialogue processing argue for a measure of coherence based on three themes: meaning, structure and intention.

We describe a theory of intention analysis for solving, in part, the problem of dialogue processing. A central principle of the theory is that coherence of natural language dialogue can be modelled by analysing sequences of people's intentions. The theory is incorporated within a computational model, called Operating System CONsultant (OSCON), implemented in Quintus Prolog, which answers, in English, English questions about computer operating systems. The theory can be used to model the level of expertise of a user. The results have implications for both the theory and engineering of natural language interfaces.

Mc Kevitt, P. (Editorial) (1992) Natural language processing. In *Artificial Intelligence Review, Vol. 6, No. 4, 327-332, Dec*, Dordrecht, The Netherlands: Kluwer-Academic Publishers.

Mc Kevitt, P. and J. Rowe (1992) An emergent computation approach to natural language processing. In *Artificial Intelligence and Cognitive Science '91*, Springer-Verlag British Computer Society Workshop Series, Sorensen, Humphrey (Ed.), 199-218. Berlin, Heidelberg: Springer-Verlag.

Mc Kevitt, P., Derek Partridge and Yorick Wilks (1992) Approaches to natural language discourse processing. In *Artificial Intelligence Review, Vol. 6, No. 4, 333-364, Dec*, Dordrecht, The Netherlands: Kluwer-Academic Publishers.

Mc Kevitt, Paul, Derek Partridge and Yorick Wilks (1992) *A survey of approaches to natural language discourse processing.* Technical Report 235, Department of Computer Science, University of Exeter, GB- EX4 4PT, Exeter, England, UK, EC.

25 May

Parallel Computers: OK for Academics - but will they fly?

Graham Nudd

Department of Computer Science

We have seen, over recent years, a substantial increase in the sophistication of real-time processing. This in part has been driven by the capabilities of the underlying technologies. However, some of the more demanding applications (including, for example, space-borne processing) have substantial system constraints. The advent of 'concurrent computing' offers the potential of performing some of the computationally intense operations with "parallel" hardware. How feasible is this and what demands does it place on the technology?

8 June

A Programmable Structured Editor

Mike Cowlishaw

IBM UK Scientific Centre
Winchester

Many sophisticated and specialised editing programs have been developed over the years. These editors help people manipulate data, but the diversity complicates rather than simplifies computer use. This talk describes an editing program that can work with the syntax and structure of the data it is presenting, yet is not restricted to just one kind of data. It is used for editing programs, documents, electronic mail, and other material, and hence provides a consistent environment for the user regardless of the editing task.

The 'live parsing' technique used by the editor means that it can be programmed to handle a very wide variety of structured data. The structure information is, in turn, used to improve the presentation of data (using colour, fonts, and formatting) which makes it easier for people to deal with the text being edited. The first implementation of the editor, the VM/CMS Live Parsing Editor (known as LEXX), has been a product in the UK and Europe for several years; its successor, LPEX, is a product for OS/2 and AIX.

The editor was originally written for the New Oxford English Dictionary project, and this will be described briefly as an introduction to the talk.

15 June

Issues and New Ideas in Computational Stereopsis

Andrew Calway

Department of Computer Science

Since its ‘discovery’ by Wheatstone in c.1838, stereopsis - the process of perceiving a 3D world from disparate binocular views - has been of continual fascination and interest, both to those wishing to understand the workings of the human visual system and to those keen to exploit it (the stereoscope, the 3D movie). Not surprisingly, it has also been the focus of extensive research within the computer vision community: given two cameras and a suitable computational model, is it possible to build up a description of the world, providing information about attributes such as depth, surface orientation and textural structure?

The purpose of this talk is twofold: to give a brief overview of the basic principles of stereopsis and summarise the progress that has been made towards a ‘binocular vision machine’; and to present the ideas and results of some new research being conducted by the speaker which aims to address some of the fundamental problems. The talk will be wide ranging and primarily aimed at the non-specialist/curious, touching on the physiological, psychological and computational aspects of the subject.