Technical Reports 1980 – 1991

University of Manchester
Department of Computer Science

August 1, 1996

Reports with numbers shown in **bold** below are available by anonymous ftp from `ftp.cs.man.ac.uk` in directory `pub/TR`, or using the WorldWide Web via URL

`http://www.cs.man.ac.uk/csonly/cstechrep/index.html`

UMCS-91-12-5 An Introduction to Categorial Grammars
Mary M. Wood

Categorial grammars come of a long tradition of linguistic description rooted in philosophy of language, logic, and algebra. Like dependency grammars, they look for the constructive patterns of semantic linkage that hold a sentence together, rather than for the analytic patterns by which it can be segmented. They are distinctive for their direct representation of semantics in syntax - which endears them to philosophers of language, and to some linguists - and for their arithmetical transparency - which endears them to logicians, and especially to formal and computational linguists. It is no coincidence that they have finally come to prominence with the dramatic rise of computational linguistics during the last ten years or less.

This introduction outlines first the most important principles and practice of the categorial enterprise. Chapter 2 offers a brief historical introduction. Chapter 3 describes one specific "benchmark" categorial grammar, the "Lambek calculus", and Chapter 4 discusses a range of different proposed extensions to that grammar.

Technical Report

UMCS-91-12-4 A Set Theoretic Semantics for First Order Temporal Logic: definition and Application using Isabelle
P. A. J. Noel

The paper contains a brief description of an exercise in using the generic theorem prover Isabelle to perform the following :

- define the semantics of a temporal logic within Zermelo-Fraenkel (ZF) set theory;

- define a number of inference rules concerning the temporal logic;
- define a tactic which uses some of the derived inference rules in backward proofs to generate automatically a model for propositional temporal formulae.

One advantage of using a theorem prover to define the semantics of a logic within another logic is that (sound) proof systems for the defined logic may be derived from its semantics. Another advantage of this approach is that it allows the derivation of prototype 'programs' (in the form of inference rules under the control of an automatic tactic) correct with respect to the semantics. An example of such a derivation is discussed in this paper.

Technical Report

UMCS-91-12-3 German Temporal Prepositions from an English Perspective
Martin Durrell and David S. Bree

The following is an attempt to analyse the procedures for selecting temporal prepositions (or conjunctions) in German according to the framework developed in Bree et al. (1990). Systematic contrast is made with the English as analysed there (and, where interesting or relevant with Dutch). The German time words follow similar selection trees for those in English and Dutch, but there are some quite interesting differences (and not a few problems), and some amendments had to be made to the analysis in Bree et al. (1990). Despite these differences, there would seem to be no reason in principle why the analytical framework in terms of decision trees adopted by Bree et al. (1990) should not be as revealing for German as for other languages.

Technical Report

UMCS-91-12-2 A Discrete Systematic Model for racing in an SR Latch
W. P. R. Mitchell and A. S. Hilditch

This paper develops a model of propagational delay for digital logic which assumes all voltages to be boolean valued. This allows for calculations on racing which can not normally be done with only boolean valued models. To illustrate this point the paper goes through a particular instance of how a latch performs meta-stably when its set and reset inputs are made 1 simultaneously, and then simultaneously made 0. The extra ingredient which generates the propagational delay is the incorporation of a value for the MOSFET SiO2/Si interface charge. That charge is then modelled as varying in a linear way over time with changes in the gate value.

Technical Report

UMCS-91-12-1 Learning a Grammar
W. Daniel Solomon

A prototype system ('L') for the induction of a lexical grammar from a sample of natural language has been produced. L takes as its input a selection of children's reading books, and induces, or learns, a Categorial Grammar, which takes the form of a lexicon, together with further corpora, to augment its grammar. L also demonstrates the validity of the grammar produced by generating further sentences from that grammar.

Technical Report

UMCS-91-11-2 Modularity in Model-Oriented Formal Specifications and its Interaction with Formal Reasoning
John S Fitzgerald

This thesis concerns the provision of facilities for the modular structuring of model-oriented formal specifications of computing systems.

The basic concepts of model-oriented specification and the need for modular structuring of specifications are reviewed. It is argued that a structuring mechanism should permit formal reasoning to proceed separately for each unit of structure. It is also argued that the provision of structuring facilities should be governed by practical experience.

The body of the thesis applies these principles to the development of a collection of structuring facilities and their application to a model- oriented specification language. A naive initial case study helps determine a basic repertoire of structuring facilities. Structured specifications will consist of definitions of modules. Each module contains an unstructured specification encapsulated by an interface which defines dependencies on, and services supplied to, other modules. Reuse of modules within and between structured specifications is supported. Abstract syntax, context conditions and denotational semantics for this system are proposed. The semantics is presented in terms of theories associated with modules and does not make reference to any particular language for the module contents. A case study using the structuring facilities developed thus far leads to a consideration of their use with a particular model-oriented language. The generation of a theory from a module's contents; the management of the name space due to the other modules in the structure and the use of state-based specifications within modules are considered. A final case study applies the techniques developed in earlier chapters. A concluding chapter assesses the extent to which the definition of structuring facilities described in the thesis has achieved its stated aims. Subjects for further research are suggested.

Ph.D. Thesis

UMCS-91-11-1 System software on the Flagship parallel graph reduction machine
K. R. Mayes, J. A. Keane and S. Holdsworth

The Flagship system was a graph reduction machine having a distributed physical architecture. Although Flagship was firmly in the declarative world, explicit state was supported in the graph reduction model to express the behaviour of the operating system. This state not only had to be isolated from the declarative aspects of the Flagship machine, but also had to be supported with respect to distribution. The mechanism provided for maintaining the consistency of state are discussed with respect to atomic actions at levels in the Flagship system. This approach is used in order to demonstrate how the software environment was supported by the basic execution mechanism of the machine. The nature, construction and creation of system software is discussed with particular reference to optimising access to distributed resources.

Technical Report

UMCS-91-7-2 Specifying Concurrent Object-based Systems using Combined Specification Notations
J. Sa and B. C. Warboys

Although many specification notations for sequential systems have been widely used in practice, most specification notations for concurrent systems, such as temporal logics, are still far too complicated to be used in practice. This report describes a specification method for concurrent object-based systems by utilising simpler notations, such as VDM, and restricted temporal logic notations. In this specification framework, specifications of systems are defined in a temporal logic. However the specifier of a system does not need to use the complicated temporal logic to specify the behaviour of the system, instead, different simpler specification notations are used to specify the different aspects into a formula defined in the temporal logic.

Technical Report

UMCS-91-7-1 An Attempt to Reason about Shared-State Concurrency in the Style of VDM
Ketil Stölen

The paper presents an attempt to develop a totally correct shared-state parallel program in the style of VDM. Programs are specified by tuples of five assertions (P,R,W,G,E). The pre-condition P, the rely-condition R and the wait-condition W describe assumptions about the environment, while the guar-condtion G and the eff-condition E characterise commitments to the implementation. The pre-, rely- and guar-conditions are closely related to the similarly named conditions in Jones' rely/guarantee method, while the eff- condition corresponds to what Jones calls the post-condition. The wait-condition is supposed to characterise the set of states in which it is safe for the implementation to be blocked; in other words, the set of states in which the implementation, when it becomes blocked, eventually will be released by the environment. The implementation is not allowed to be blocked during the execution of an atomic statement. Auxiliary variables are introduced to increase the expressiveness. They are used both as a specification tool; to characterise a program that has not yet been implemented, and as a verification tool; to show that a given algorithm satisfies a specific property. However, although it is possible to define history-variables in this approach, the auxiliary variables may be of any type, and it is up to the user to define the auxiliary structure he prefers. Moreover, the auxiliary structure is only a part of the logic. This means that auxiliary variables do not have to be implemented as if they were ordinary programming variables.

Technical Report

UMCS-91-6-2 Proposal for an Information Model for EDIF
Rachel Lau

This report describes a partial information model of EDIF 2 0 0. First of all, an outline of the purpose of an information model for EDIF is presented. Then, a general overview of the model is given, followed by a more detailed description of the modelling work. The complete dictionaries of terms are given in the following chapters and an index of terms is included at the end of the report. The model itself which consists of a total of 17 schemas can be found in appendices A-Q. The four appendices R-U identify respectively : the point of definition of an EXPRESS object within the schemas; whether or not an EDIF construct has been modelled; the available attributes for each entity; and the relationships in which an object may occur.

Ph.D. Thesis

UMCS-91-6-1 A Method for the Development of Totally Correct Shared-State Parallel Programs
Ketil Stölen

A syntax-directed formal system for the development of totally correct programs with respect to an (unfair) shared-state parallel programming language is proposed. The programming language is basically a while- language extended with parallel- and await-constructs. The system is called LSP (Logic of Specified Programs) and can be seen of as an extension of Jones' rely/guarantee method. His method is strengthened in two respects :

- Specifications are extended with a wait-condition to allow for the development of programs whose correctness depends upon synchronisation.
- Auxiliary variables are introduced to increase expressiveness. They are either used as a specification tool to eliminate undesirable implementations or as a verification tool to prove that a certain program satisfies a particular specification.

Technical Report

UMCS-91-5-1 Interference Resumed
C. B. Jones

Compositional development methods for concurrent (interfering) programs are harder to find than for sequential (isolated) programs. Rely and guarantee conditions extended operation decomposition methods for sequential programs to cover concurrent shared-variable systems: interference was specified in order to achieve compositionality. Ketil St¨{o}len's thesis addresses shortcomings of the rely/guarantee idea: synchronization is specified by a wait-condition and expressiveness is increased by judicious use of auxiliary variables. This paper describes a new step in the quest for compositional development methods for concurrent programs. A collection of temporal operators aimed at natural reasoning about programs is indicated; these operators apply to predicates of two states and a style of specification is proposed which clearly separates transitions of a component from those of its environment. Furthermore a case is made for 'resumptions' as a semantic model for the logic. Since this paper reports on work-in-progress, a list of known problems is given.

Technical Report

UMCS-91-4-3 The HDG-Machine: A Highly Distributed Graph-Reducer for a Transputer Network
Hugh Kingdon, David R. Lester and Geoffrey L. Burn

Distributed implementations of programming languages with implicit parallelism hold out the prospect that the programs are immediately scalable. This paper presents some of the results of our part of Esprit 415, in which we considered the implementation of lazy functional programming languages on distributed architectures. A compiler and abstract machine were designed to achieve this goal. The abstract parallel machine was formally specified, using Miranda (trade mark of Research Software). Each instruction of the abstract machine was then implemented as a macro in the Transputer Assembler. Although macro expansion of the code results in non-optimal code generation, use of the Miranda specification makes it possible to validate the compiler before the Transputer code is generated. The hardware currently available consists of five T800-25's, each board having 16M bytes of memory. Benchmark timings using this hardware are given. In spite of the straight forward code-generation, the resulting system compares favourably with more sophisticated sequential implementations, such as that of LML.

Technical Report

UMCS-91-4-2 A Performance Assessment of the GSM Radio Link Protocol (RLP)
Brian W. Marsden

his paper presents the results of a simulation evaluation of the Radio Link Protocol (RLP), which is an adaptation of HDLC for use on radio links in cellular networks. The RLP protocol is first described, in particular the ways in which it differs from standard HDLC, in the context of the special characteristics of radio links. The simulation model is then discussed, with particular emphasis on the choice of operational parameters and error recovery policies, which are crucial issues given a radio link which is inherently unreliable and contributes a propagation delay considerably in excess of the time taken to transmit a frame. The results of the study are then presented, with recommendations of suitable operational parameter values and some suggested changes to the RLP specification. The central aim of this work is to determine the combination of operational parameters which will maximise throughput and minimise transit time for a useful range of link error rates.

Technical Report

UMCS-91-4-1 An Algorithm for Planning 'Sensible' Routes
Ian Pratt

The problem of finding an optimal route between two points in a space strewn with obstacles has been extensively studied in robotics. The standard approach to such problems is first to reduce the original geometrical specification to a finite graph of possible routes, and then to use graph-searching techniques together with the metrical information available in the original geometrical specification to select the best of these routes. The algorithm presented here, by contrast, employs a 'first-past-the-post' approach to route planning. Specifically, we show that the algorithm is guaranteed to produce a route satisfying a certain criterion of 'sensibleness', because routes failing this criterion are always slower to get constructed. The approach taken here differs in three respects from most current work on route planning:

- it finds sensible, but not necessarily optimal, routes;
- it is uninfluenced by perspective (though it is limited by occlusion problems);
- it produces its output in a qualitative form, which is nevertheless well-suited to guide the appropriate route- following behaviour.

Technical Report

UMCS-91-3-3 Branching Time and Partial Order in Temporal Logics
Wojciech Penczek

The aim of this report is to present existing formal languages of propositional temporal logic with frames based on branching time structures or, more general, partial orders. Branching time and partial order logics do not only differ in their underlying frames, but also in the way the logics are linked to the behaviour of concurrent systems.

Technical Report

UMCS-91-3-2 Distributed Algorithms for Detecting Distributed Deadlock
Steve Hilditch and Tom Thomson

We present two simple algorithms for the detection of deadlock on a distributed database system. They are designed to be scalable and minimise network message traffic. An object-oriented approach is used and semi-formal proofs of the algorithms' correctness are given.

Technical Report

UMCS-91-3-1 A Distributed Termination Detection Scheme
Xinfeng Ye and John A. Keane

A fully distributed scheme for detecting the termination of distributed computations is proposed. The scheme does not require a pre-defined detector, and takes into account problems such as network delay and the non-order-preserving arrival of messages. The scheme can be applied to any kind of connection topology. The correctness of the scheme is presented in terms of showing that the global stable condition holds when the scheme declares the termination of the computation. The upper bound of the number of the messages which are used to detect termination is also discussed.

Technical Report

UMCS-91-2-2 Operational Semantics for Hardware Design Languages
Howard Barringer, Graham Gough and Brian Monahan

An operational semantics for two hardware design languages, abstracted from Abel and ELLA, are given using Structural Operational Semantics. This can be used to calculate a state machine representation suitable for verification purposes. We show how, despite differences in semantic structure, the languages discussed possess various common features, such as the use of iterative fixpoints to calculate stable states under value propagation. Although hardware designers expect their designs to represent finite state machines, explicit semantics is needed to calculate these state machines in general. This enables construction of a reference simulator against which output of other, more efficient simulators can be compared.

Technical Report

UMCS-91-2-1 Distributed Simulation Using "Relaxed Timing"
Alvaro Garcia Neto

Research in computer architecture relies increasingly in simulation as opposed to prototyping. However, as the size and complexity of the simulated systems increases, "conventional" sequential simulation fails to deliver the necessary performance. We identify the requirement to model time accurately as a main cause for poor simulation performance. We propose a new simulation philosophy, Relaxed Timing, in which the emphasis is on speed of execution rather than correct modelling of time. In this philosophy, modules are entirely event-driven, and consume data as soon as they become available – no synchronisation is done to ensure timing accuracy. Because performance and accurate timemodelling are conflicting requirements, relaxation of temporal precision yields higher performance. This approach clearly leads to preemption errors. We show that, for "self-scheduling" and "well balanced" systems, such as dataflow, these errors are acceptable. We show that a relaxed timing dataflow simulator performs better than a sequential simulator for a large number of algorithms and significant range of input data. Having shown the adequacy of the model, we investigate possible improvements, and find that the insertion of time-priority queues minimizes preemption. By blocking these queues at strategic times, a spectrum of timing precision is possible. We implement and discuss this enhancement. The development language for the simulator is occam. We assess the suitability of occam as a specification language for the generic simulation of computer architectures, and find it adequate. We also discuss the need for tools of higher levels of abstraction to ease the task of simulating distributed parallel systems, and place this need in the context of current research in programming environments, software engineering and program visualization. We describe the available tools in the hierarchy of abstraction levels, and argue that occam is a good candidate to fill an existing gap in this hierarchy.

Ph.D. Thesis

UMCS-91-1-1 Development of Parallel Programs on Shared Data-Structures
Ketil Stölen

A syntax-directed formal system for the development of totally correct programs with respect to an unfair shared-state parallel while- language is proposed. The system can be understood as a compositional reformulation of the Owicki/Gries method for verification of parallel programs. Auxiliary variables may be of any sort, and it is up to the user to define the auxiliary structure he prefers. Moreover, the auxiliary structure is only a part of the logic. This means that auxiliary variables do not have to be implemented as if they were ordinary programming variables. The system is proved sound and relatively complete with respect to an operational semantics and employed to develop three nontrivial algorithms: the Dining-Philosophers, the Bubble-Lattice-Sort and the Set-Partition algorithms. Finally, a related method for the development of (possibly nonterminating) programs with respect to four properties is described. This approach is then used to develop Dekker's algorithm.

Ph.D. Thesis

UMCS-90-10-1 Indexed Categories for Program Development
B. Hilken and D. E. Rydeheard

We consider indexed categories as an algebraic framework and meta- language for program logics. The logics incorporate programs in a programming language, conversions between programs, specifications of program behaviour in a specification language, and explicit proofs of program correctness. Program development methods are interpreted as constructions of indexed categories. Two examples are given: (1) type invariants and (2) specifications of operations and their implementation.

Technical Report

UMCS-90-9-1 The Random Matrix Hashing Algorithm
Behnam Banieqbal and Steve Hilditch

The Random Matrix hashing algorithm is a new hashing algorithm which has the following properties :

- It can be sized to any bit length input and output.

- It has scattering properties which are indistinguishable from the theoretical best. It will out-perform the industry standard hashing functions, e.g. dividing by a prime number, dividing by the Golden Ratio.
- It is a source of an infinite number of different hashing functions.
- It can be implemented in hardware to hash most inputs in one clock cycle. The number of gate delays required is proportional to the logarithm of the number of input bits. It can also be pipelined.
- It can be chosen to be invertible if required.

The algorithm consists of multiplication of the input, thought of as a binary row vector, by a fixed random binary matrix, the result is a binary column vector, the hashed value. We also derive formulae of the theoretical performance of theoretically best, completely random hashing functions. We include graphs to show that our algorithm performs as the theoretical best on substantial test data.

Technical Report

UMCS-90-8-2 Distributed Binding Mechanisms in the Flagship System
J. A. Keane

The Flagship machine is a parallel, graph reduction machine designed to execute functional languages. The machine supports a global address space across its distributed physical store. A program to run on the machine is compiled into a graph of packets and the graph is then reduced. The load balancing of the machine is entirely dynamic and consequently packets must be moved and copied at run-time. This paper considers the dynamic mapping of the graph of packets to the distributed stores that make up the Flagship machine. In particular, the dynamic mapping of packets across the machine is viewed in a binding framework. For high performance in a machine such as Flagship a large proportion of memory accesses should be localised to reduce communication overheads. There are two binding mechanisms in the machine that provide global-to-local address packet mapping to achieve high performance: a conventional cache mechanism and a static copying mechanism. The paper primarily deals with the motivation for the static copying mechanism, and considers the use and management of the mechanism by the system software of the machine.

Technical Report

UMCS-90-8-1 Distributed Relational Queries: Structures for Locking and Access
Steve Hilditch and Tom Thomson

We consider scheduling relational database queries on loosely-coupled multi processors, the implementations of both two-phase locking and index structures, especially secondary indexes. We consider resources which can be arranged in the form of a directed graph and we assume that resources which are close in the hierarchy are close in the processor site network. A strategy for efficient distributed locking and releasing of hierarchically arranged resources is given together with a discussion of its costs. We also present a novel implementation of secondary indexes on loosely-coupled multiprocessors, making use of the natural location of resources. Apart from the usual constraint that concurrency be maximised, we consider that careful note should be taken of both :

- the number of locks needed to be taken.
- the number of network messages that need to be sent.

Our idea is to use possible hierarchical structures and reasonable resource loading to reduce the number of locks taken and the number of network messages needed to implement two-phase locking. Examples of our algorithms' usefulness to manage relational database queries are given.

Technical Report

UMCS-90-3-1 On the usability of logics which handle partial functions
J. H. Cheng and C. B. Jones

Partial functions are common in computing – they arise from recursive definitions and from the basic operators of data types like sequences. The need to prove results about logical formulae which include terms using partial functions must cast doubt on the applicability of classical logic. Approaches to handling partial functions which attempt to salvage classical logic include avoiding function application and the use of variant equality notions. The alternative is to accept the need for a non-classical logic. The most frequently used non-classical logic adopts non-strict (but also non-communicative) conditional versions of the propositional operators. An alternative – which is more fully explored below – uses commutative and/or operators as defined in the truth-tables which Kleene attributes to Lukasiewicz. A consistent and complete proof system for such a logic has been used in the program development method known as VDM. An important feature of this approach is the way the definitions of partial functions are brought into proofs. This paper analyzes different approaches to reasoning about partial functions. In particular, it compares the usability of different logics in handling the sort of partial functions which arise in the specification and design of computer software.

Technical Report

UMCS-90-2-1 The Flagship Declarative System
J. A. Keane

The Flagship Project was a research collaboration between the University of Manchester, Imperial College, London and International Computers Limited. Its aim was to produce a complete computing system based on a declarative programming style. This survey report discusses the approaches taken in the three major areas addressed by the project: (1) programming languages and programming environments, (2) the machine architecture and computational models and (3) the software environment, and, where appropriate, presents the approaches with regard to general classifications. The influences on the approaches taken by Flagship are considered and other similar approaches are briefly discussed. The intention is to present the different areas of the project both as a coherent whole and in comparison to similar activities. The work in areas (2) and (3), which involved the groups at the University of Manchester, is concentrated upon. Some discussion of the work in area (1), and thus the contribution of the Imperial College part of the project, is also included.

Technical Report

UMCS-90-1-1 Modularizing the Formal Description of a Database System
John S. Fitzgerald and Cliff B Jones

Specification languages have been proposed, and are being developed, which offer ways of splitting specifications into separate components or modules. It is important that such languages are able to cope with modularizations which are required by realistic specification tasks. This paper offers a challenge problem in modularization based on a description of an existing database system. The chosen modularization is motivated by the need to separate coherent units about which useful properties can be deduced.

Technical Report

UMCS-89-12-1 Specifications are not (necessarily) executable
Ian J. Hayes and Cliff B. Jones

Specifications can be written in languages which have formal semantics. Their very formality, and the similarities with some aspects of implementation languages, invites the idea that specifications might be executed. This paper presents a number of arguments against that idea. The aim is to warn of the dangers of limiting specification languages to the point where all of their constructs can be executed. While conceding the difficulties of relating specifications to an understanding of the "requirements" for a system, it is argued that other solutions should be sought than "executable specification languages".

Technical Report

UMCS-89-11-2 Concurrency Control in the Multi-Ring Manchester Dataflow Machine
Yong Meng Teo

Extensive research effort has been spent on developing parallelising compilers that explicitly extract parallelism in programs coded in conventional programming languages. Recent research has focused on the design of new languages, such as functional and single-assignment languages, which have inherent program parallelism. These research efforts have a common objective of exposing maximum program parallelism for exploitation in parallel computer systems so as to attain higher computational performance. However, over-abundant program parallelism, when left uncontrolled, runs the risk of the machine getting swamped in parallelism resulting in excessive (wasteful) usage of memory resources.

The aim of this thesis is to demonstrate that program parallelism can be effectively controlled so as to fully exploit the available machine parallelism. Using the Multi-Ring Manchester Dataflow Machine as an example, an architectural approach to program concurrency control in this machine is proposed. This approach tackles parallelism control at two levels. Inter-process parallelism is controlled by a process-based hardware throttle which imposes depth-first process execution. The throttle is a self- adaptive feedback control mechanism that matches program and machine parallelism by scheduling processes for execution depending on the machine activity level. Intra-process parallelism is restrained by controlling the execution of iterative instructions. Two approaches to controlling iterative instructions, the self-scheduling method and the process-tree method, are studied. Apart from reducing memory usage, controlling iterative instructions has the important effect of reducing the occurrence of pipeline holes caused by lengthy iterative instructions.

Simulation of the extended dataflow system with parallelism control has produced encouraging results, which are analysed in some detail. Major problems of parallelism control are exposed and some solutions to these problems are discussed. In conclusion, the optimal multi-ring configuration that can be supported by a hardware throttle is established. This research has shown that the proposed prallelism control strategy is highly effective in reducing memory resource usage and refuting the criticism that programs executed in parallel systems require substantially more memory resources.

Ph.D. Thesis

UMCS-89-11-1 Manchester Dataflow Machine: Benchmark Test Evaluation Report
John Foley

ABSTRACT TO GO IN HERE – JRG to provide

Technical Report

UMCS-89-10-6 Characterising Temporal Logic
Michael Fisher

This report extends some of the previous work on the characterisation of modal and tense logics and develops a framework suitable for the characterisation of simple properties of temporal. In particular, characteristic temporal formulas are given for linear, dense, and infinite temporal model structures, using only the U+ temporal operator.

Technical Report

UMCS-89-10-5 Colour System for Smalltalk-80
Trevor P. Hopkins

This document describes an experimental enhancement to the Smalltalk-80 system which supports colour displays. In summary, the improved colour system allows all output (including text) to be made using colour. It provides several new classes, including ColorForm and ColorDisplayScreen. These provide the same functionality as the standard classes Form and DisplayScreen, except that additional colour operations are supported. A new global variable ColorDisplay (a reference to an instance of ColourDisplayScreen) replaces Display for colour output. A class ColorDevice is provided to represent the colour display hardware.

It should be stressed that the system described only works with ParcPlace Smalltalk V12.3, on Sun 3/60 workstations. It may be possible to get this system to work using other versions of ParcPlace Smalltalk, or other hardware platforms, but has not yet been tried. Unless otherwise stated, remarks herein refer to release 1.01 of the colour system.

Technical Report

UMCS-89-10-4 MetateM: A Framework for Programming in Temporal Logic
Howard Barringer, Michael Fisher, Dov Gabbay, Graham Gough and Richard Owens

In this paper we further develop the methodology of temporal logic as an executable imperative language, presented by Moszkowski and Gabbay and present a concrete framework, called METATEM for executing (modal and) temporal logics. Our approach is illustrated by the development of an execution mechanism for a propositional temporal logic and for a restricted first order temporal logic.

Technical Report

UMCS-89-10-3 Operation Decomposition Proof Obligations for Blocks and Procedures
Jean Alain Ah-Kee

In the context of a compositional development method, operation decomposition proof obligations are given for a language with blocks, arrays, recursive procedures, unrestricted procedure calls with value, value-result, reference parameters and global variables. Post-conditions of two states relating the initial and final states of the computation are used; existing proof rules for languages with the above features are for post-conditions which are predicates of the final state only.

Aliasing, as arising from reference parameters, and static scoping are dealt with in the Hoare-like proof system by the use of a “syntactic environment” (in a similar way as is usually done in denotational semantics).

A denotational semantics is given to the language. The proof rules are proven to be consistent with the semantics and are also proven to be (relatively) complete. The denotational semantic model uses an environment which includes a set giving all the locations in scope, and procedure denotations carry with them the locations in their “support”. The use of such an environment, as opposed to a more standard one in which information about the locations in scope is not included, makes easier the proofs of consistency and completeness for the proof rules given. The semantic model using the extended environment also has some interesting properties. It is shown to be congruent with a semantic model using a more standard environment.

In the semantic model, locations are allocated in a non deterministic way to variable declarations. This is shown to add no real non-determinism to the language.

Ph.D. Thesis

UMCS-89-10-2 Fair SMG and Linear Time Model Checking
Howard Barringer, Michael D. Fisher and Graham D. Gough

SMG is a system designed to generate a finite state model of a program from the program itself and an operational semantics for the programming language. Model-checking can then be used to verify that the program satisfies a set of desired temporal properties.

In this paper we first show how we have incorporated notions of fairness into SMG; in particular, a user is now able to define semantics with "fair" constructs, for example, parallel composition, repetetive choice, etc. A program may contain several fair constructs, each with its own fairness type. Secondly we describe a practical approach to model checking of linear temporal formulae over the fair structures generated by the new SMGF. Our approach is a refinement and extension of the fair-satisfiability algorithms, presented earlier by Lichtenstein and Pnueli, together with techniques developed in our practical implementations of decision procedures for linear temporal logic.

Technical Report

UMCS-89-10-1 Decision Procedures for Temporal Logic
Graham D. Gough

Temporal Logic has been shown, to be a powerful tool for reasoning about concurrent programs. A decision procedure is an algorithm for determining whether a given formula of the logic is valid (i.e. it is true in all interpretations of the logic). This dissertation is a study of decision procedures for various forms of temporal logic; in particular an improved decision procedure is given for a future time Temporal Logic, and a new decision procedure is given for a form of Temporal Logic including past time operators.

M.Sc. Thesis

UMCS-89-9-2 Some Special Cases of Linear Approximation of Symmetric Matrices
K.K. Lau and K. Zietak

We consider the linear problem of best approximating a matrix by elements of a subspace in the normed linear space of symmetric matrices. First we identify some special cases for the spectral norm by considering the dual problem. Then we generalise the results and consider arbitrary unitarily invariant norms, and identify a further special case.

Technical Report

UMCS-89-9-1 Multicast Facilities for Multimedia Group Communication Environments
Lek H. Ngoh

Complex distributed messaging systems are being developed rapidly to take advantage of computing environments which consist of high-performance workstations and high-speed local area networks. The Distributed Multimedia Information Systems (DMISs) proposed heresupport the exchange of multimedia information between a group of users. The architectural design of a DMIS is presented in this thesis, together with an investigation, design, implementation and evaluation of a set of multicast facilities which support the operations of systems such as a DMIS.

This thesis shows that multicast communication facilities are essential in providing an effective group communication environment. A multicast model which satisfies the various requirements and characteristics of systems such as DMIS, is proposed. To obtain a better understanding of multicast, a prototype implementation based on the proposed model is carried out. This is followed by an initial performance evaluation using a set of experiments. The results reveal areas of the prototype implementation which can be refined to increase performance. The improved performance of the refined version is shown experimentally. Some future directions of this research are also discussed.

Ph.D. Thesis

UMCS-89-8-3 Categorical ML – Category-Theoretic Modular Programming
Esther Dennis-Jones and David E. Rydeheard

We consider an extension of the functional programming language Standard ML with a modular structure based upon concepts in category theory such as categories, functors, natural transformations and adjunctions. In essence, what we are doing is following the categorical imperative of considering arrows as well as objects. This is intended to enforce a certain mathematical rigour on the programmer, so that the only programs that can be expressed are those with a categorical significance. Because of the algebraic nature of category theory, we may generate equational correctness conditions for the modular structure of programs, thus separating the correctness of individual functions from that of modules. We describe this programming language, give examples of its use, and explain how it is implemented in a type system.

Technical Report

UMCS-89-8-2 Spatial Reasoning and Route Planning using Sinusoidal Transforms
Ian Pratt

This paper investigates a method of spatial representation whereby convex regions of the plane are transformed into periodic functions of one variable. The resulting representation scheme – called the sinusoidal representation scheme – is shown to facilitate a variety of commonsense spatial reasoning tasks. A guiding principle behind the system presented here is that it should perform well over a limited repertoire of spatial reasoning problems, whilst making the least possible use of explicit mathematical machinery. Thus, although it does not aspire to model human commonsense spatial reasoning, the system is conceived of as potentially playing the same role in a robot as that played by commonsense spatial reasoning in humans. The final section of the paper applies the sinusoidal representation scheme to the problem of route planning for a point moving in a plane strewn with convex obstacles. A method is developed to identify not the precise set of optimum paths, but rather, a set of sensible paths guaranteed to contain all such optima.

Technical Report

UMCS-89-8-1 An Investigation into Architectures for a Parallel Packet Reduction Machine
Mark Irvine Greenberg

Declarative programming languages show promise in solving two fundamental issues in Computer Science, that of providing concise reliable software, and a natural methodology for programming parallel computers.

The Flagship project is a collaborative academic and industrial venture, which aims to produce a complete parallel computer system capable of supporting declarative languages efficiently. A packet-based reduction model of computation has been specifically developed for parallel execution and to support the features of declarative languages. The research described in this thesis examines the machine structures and the organisation needed for efficient packet reduction. Two main areas are covered, the architecture of the reduction processor, and the network to interconnect the processors.

A high performance delta network is proposed for the reduction machine. The individual switching elements employ a hybrid circuit/message switching technique in conjunction with message buffers to achieve performance. The switching elements also have the capability to distribute work dynamically to processors of the machine, in accordance with workload information. The design of the hybrid switch is described and simulation results are presented to substantiate the claims for efficiency. A number of experiments to investigate performance under varying network conditions are also described.

The processor investigation examines three candidate architectures to support packet reduction. These architectures range from a sophisticated quad processor architecture where each processing action is carried out by a separate unit, to a single processor architecture where all processing functions are implemented by one unit. The architectures are simulated at a detailed level to model the effects of pipelining and contention, and to monitor the utilisation of the units in the processor. The results from simulating a suite of benchmark programs indicate that a dual processor architecture is the most effective in terms of performance and hardware costs. The simulated execution times are compared to the ambitious performance goals of Flagship, and the conclusion reached is that the target performance can be met through the use of the emerging generation of reduced instruction set microprocessor technologies.

Ph.D. Thesis

UMCS-89-7-1 Symbolic Execution as a Tool for Validation of Specifications
Ralf Kneuper

This thesis investigates symbolic execution, a method originally developed for the verification and validation of programs, and extends it to deal with (state-based) specification languages as well.

The main problems encountered are implicit specifications and the intended generacity with respect to the specification language used. Implicit specifications are handled by introducing 'description values': identifiers take as values predicates which describe their 'actual value', the value one would get by actual execution.

Language genericity is achieved by expressing the definition of symbolic execution in terms of the semantics of the specification language used. Both denotational and operational semantics of symbolic execution are discussed. The denotational semantics of symbolic execution, expressed in terms of the denotational semantics of the specification language, provide a correctness notion for symbolic execution. A description of the operational semantics of the language is used as a parameter to tailor symbolic execution to that language.

Based on these ideas, a symbolic execution system called SYMBEX is described and specified. SYMBEX is to form part of the project support environment IPSE 2.5 and help the user to validate a specification by symbolically executing it.

Finally, the achievements and limitations of this approach are discussed, and suggestions made for future work.

Ph.D. Thesis

UMCS-89-6-1 Distributed Deadlock Detection: Algorithms and Proofs
Steve Hilditch and Tom Thomson

We consider loosely-coupled multiprocessors on which memory is distributed. We discuss transaction schedulers which both serialise transactions and minimize network messages.

Two distributed deadlock detection algorithms are given which are 'truly distributed' in the sense that they require no synchronisation procedures or central control. These algorithms can be used in the more general distributed case. Formal proofs of their correctness are given, together with a discussion of their efficiency and a comparison with published algorithms.

We believe that the formal proof notation and method should be useful for verifying other distributed algorithms where communication is by message passing.

Technical Report

UMCS-89-3-1 The Heterogeneous Multi-Ring Dataflow Machine Simulator
A.P.W. Bohm and Y.M. Teo

Increased complexity in parallel and multiprocessor system architectures has accentuated the need for tools that assist in understanding the behaviour and evaluating the performance of these systems. This report describes a simulation model of the proposed. Heterogeneous Multi-Ring Dataflow Machine (MDM), consisting of a number of processing element rings, structure store rings and one global allocator ring. The structure of the multi-ring dataflow machine simulator, MR, is presented. Distributed storage management is introduced. A proposed dynamic hardware throttle mechanism which matches program parallelism to machine parallelism is discussed. A scheme that controls the execution of macro dataflow instructions is proposed for integration into the throttle mechanism. This report serves as the defining document for MR, which is one of the tools used for programming and architectural experimentations in the proposed multi- ring Manchester Dataflow Machine.

Technical Report

UMCS-89-2-1 A Categorial Syntax for Coordinate Constructions
Mary McGee Wood

A generalised categorial grammar is proposed for a wide range of coordinate constructions, principally but not exclusively English, which attempts systematically to exploit the use of Lambek's 'product' connective. A historical outline of categorial grammars is followed by consideration of the formal space of possibilities in terms of the set of operations permitted within a particular grammar: the rules of the Lambek calculus and some later proposals are individually discussed and their use in syntactic description illustrated.

The accounts of coordination offered within a number of alternative linguistic theories are discussed (Generalized Phrase Structure Grammar, Lexical-Functional Grammar, Word Grammar), as are previous categorial accounts (Geach, Steedman, Dowty, Oehrle, and others).

An original categorial syntax for coordinate constructions is then proposed, based on the use of the Lambek 'product', a connective forming ordered pairs, which has been neglected or avoided in previous accounts. The category assignment for a coordinating conjunction in English is XI(X+*), an infix operator returning a single entity of any category from a string of entities of that category. On this basis a detailed grammar is offered for the coordination of canonical "constituents", of "non-constituents", and for gapping, where I proposed recovery by "reconstitution" of the category of the gap. The use of products is shown to eliminate the need for the undesirably powerful rules of generalized and disharmonic composition and of unrestricted type-raising which characterize otherwise similar current work. Multiple conjunctions, coordination of unlike categories, and subordinating conjunction are also discussed.

In conclusion I discuss more generally the nature, problems, and prospects of a categorial grammar incorporating products, and the significance of formal "equivalence".

Ph.D. Thesis

UMCS-88-12-2 SMACK: The Smalltalk Actor Kernel
Trevor P. Hopkins

There has been increasing interest in recent years in concurrent object-oriented systems: systems in which at least some of the objects can be 'active' concurrently, and (typically) can continue processing after a message send. The aim is to be able to express algorithms and applications in a concurrent manner which can be exploited by aparallel computer system.

This report considers 'actors' systems, and describes an implementation of an actor computational model. It further discusses various implementation techniques for actor programming languages on realistic large-scale parallel computers.

Technical Report

UMCS-88-12-1 Review of Concurrent Object Systems
Trevor P. Hopkins

There has been increasing interest in using the 'object' paradigm to exploit concurrent algorithms on parallel computers. This report reviews some of the published work in this area. This report goes on to consider in more detail the 'actors' model for concurrent objects, and investigates the problems of implementing such systems.

Technical Report

UMCS-88-10-1 Synthesis of Logic Programs for Recursive Sorting Algorithms
K.K. Lau and S.D. Prestwich

We describe the application of folding strategies described in an earlier report to the synthesis of procedures sets for selected well-known recursive sorting algorithms. Complete logic programs which define these algorithms are presented.

Technical Report

UMCS-88-8-2 The Use of Knowledge Based Techniques For Electronic Computer Aided Design
N. P. Filer

Electronic Computer Aided Design (ECAD) is described as a process where the use of computers is required because of the complexity of the objects being designed. It is known that computer methods used in ECAD problems do not always produce acceptable results. In the past however, it was frequently possible to complete a design manually, after a computer had done all it could.

With the increasing use of Very Large Scale Integrated (VLSI) designs it is no longer possible to rely on the designer to complete the design process. Therefore, design techniques must be adopted which improve the behaviour of computerised design tools.

The use of expert human knowledge, encoded as rules and used directly by a computer whilst carrying out design tasks, is gaining popularity. Similar systems have been used successfully in other domains, such as medicine.

A methodology for using expert knowledge alongside applicative design processes and an appropriate support environment, PESL, have been developed by the author. The rationale behind the design of PESL and the main features of the PESL language and environment are described. The use of PESL has been demonstrated by a set of prototype applications developed both by the author and by colleagues. These applications are described and critically evaluated.

The development of more example systems is suggested, in order to show the methodology being used for a number of other ECAD tasks.

Ph.D. Thesis

UMCS-88-8-1 Recursion Transformations for Run-Time Control of Parallel Computations
V.J. Bush

There is much current research interest in parallel computer architectures for general purpose, high-speed computation. Part of this research effort has been devoted to the study of dataflow computer systems in which programs are represented by directed graphs, with nodes representing operators and arcs representing data dependencies between nodes. Several such computer systems have been constructed and are now operational. These include the prototype dataflow machine which has been running at the University of Manchester for the past six years.

Experience of programming this machine has been gained, and guidelines for algorithm design are becoming apparent. In particular, development of suitable parallel algorithms is difficult for the programmer because the conditions prevailing when the program runs may vary between systems, and even during a single run. In considering the nature of an appropriate development methodology for such programs it seems that the fitting of algorithm to machine structure should be an automated process, rather than the responsibility of the programmer. It has become clear that transformation of algorithms will be of paramount importance in achieving an automatic match with the machine configuration.

The aim of the research reported in this thesis is to build an algebraic reasoning system for dataflow. A functional language is used as a high level notation for dataflow graphs. This facilitates reasoning about graphs and, in particular, it enables the development of transformations between different graphs. An operational semantics for a (dynamic) dataflow model is described and related to an algebraic semantics for the functional language. A taxonomy of recursive forms of algorithms is presented and the relationship between the form of an algorithm and its operational behaviour is discussed. Transformations between different recursive forms are studied and new transformations between sequential and parallel recursive forms, which are applicable at run-time as well as at compile-time, are developed. Examples of using these transformations are discussed.

Ph.D. Thesis

UMCS-88-7-1 On Computing Best Symmetric Matrix Approximations
K.K. Lau

We consider the problem of computing best approximations in the linear space of symmetric matrices, endowed with the spectral norm, by elements of linear subspaces. We show how solutions to the dual problem can be used for computing such best approximations in general, and identify some special cases of the problem and discuss the details of computing the corresponding best approximations by making use of the dual wherever appropriate.

Technical Report

UMCS-88-6-2 The Joint Academic Network JANET
Manjula Patel

The Joint Academic Network (JANET) came into being on the 1st April 1984. It represents the rationalisation of previously diverse communications in the research and academic communities. This report examines the evolution of JANET, and the communications protocols it uses; in particular X.25 and the Coloured Books, and their relationship to the seven layered OSI Model. Taking the University of Manchester Regional Computer Centre (UMRCC) to be a typical Network Operations Centre, the internal workings of JANET are also explored.

Technical Report

UMCS-88-6-1 Semantics of Object-Oriented Languages
Mario I. Wolczko

Object-oriented programming is becoming an important technique in the construction of large software systems. Compelling arguments, like reduced maintenance costs, are advanced to encourage its use. To maximise the advantages of such methods, object-oriented programming languages need to be well-designed. When selecting the main features of a programming language, or choosing between alternative designs, formal methods or semantic analysis are invaluable. To date little attention has been given to the formal description of object-oriented languages. This thesis introduces a framework for describing the semantics of object-oriented languages.

To characterise the important features of object-oriented languages, an idealised object-oriented language is described and its semantics specified formally, using the denotational style of VDM. Design alternatives are explored in the same way. Several general principles of object-oriented language design are introduced, and the alternatives reviewed in the light of these principles. By choosing apposite semantic domains, the fundamental concepts of object-oriented systems are exposed: two message-passing schemes, based on dynamically-bound procedure call and delegation, are presented; class- and prototyped-based systems are described; and special emphasis is given to the different approaches to class-based multiple inheritance. The encapsulation of behaviour within classes is discussed, and suggestions are made as to how this might be best achieved. A variety of object-oriented control mechanisms are also surveyed.

Ph.D. Thesis

UMCS-88-4-1 Temporal Term Rewriting
William P.R. Mitchell

In this paper we explore an extension of term rewriting , temporal term rewriting, which uses temporal logic formulae to restrain the order in which rewrite rules occur in a reduction sequence. We extend the usual ideas (local confluence etc.) to temporal rewrite systems and prove that 'forgetful' terminating temporal systems are confluent if they are locally confluent. Then we prove that if a system is 'derived', it is decidable whether it is locally confluent.

Technical Report

UMCS-88-2-1 Synthesis of Recursive Logic Procedures by Top-down Folding
K.K. Lau and S.D. Prestwich

In a logic programming framework, an algorithm can be represented by a logic program; in particular, a recursive algorithm will correspond to a program containing recursive procedures. In this report, we describe some strategies for synthesising recursive logic procedures from their logic specifications, which use the standard technique of folding for recursion introduction in a top-down manner.

Technical Report

UMCS-87-12-9 Teaching Notes for Systematic Software Development Using VDM
Cliff B. Jones

This is the second version of the "Teaching Notes" for the 1986, Prentice- Hall International publication: "Systematic Software Development using VDM". It is still the case that not all of the planned material has been included (but the notes are somewhat more complete than in UMCS-86-4-2). This version is also laid out with LaTeX thus making it easier to relate to the original book. The author would be grateful for feedback both on errors and proposals for extensions.

Technical Report

UMCS-87-12-8 An Experimental User Interface for a Theorem Proving Assistant
Cliff B. Jones and R. Moore

A Theorem Proving Assistant is taken to be a computer program which can be used interactively to keep track of, and automate some steps of, attempts to prove theorems. This paper describes an experiment in the design of user interfaces to such programs. The approach is introduced using a formal description from which the program has been implemented in Smalltalk-80.

Technical Report

UMCS-87-12-7 Logical Frames for Interactive Theorem Proving
Peter A. Lindsay

This report addresses theoretical issues to do with providing mechanical support for formal reasoning independent of particular logics. More precisely, the aim is to provide support for a class of logics, identified for their utility in Formal Methods of software engineering. The main problem is to formalize – and indicate what support is necessary for – the logical frame, by defining the notions "well-formed formula", "inference rule", "proof", "theory", "logic" and so on. The emphasis is on systems suitable for mechanization as interactive computer programs. We survey various approaches to the problem, including different type theories which might serve as logical frames. Our proposal for the FRIPSE logical frame is presented in the working paper.

Technical Report

UMCS-87-12-6 Symbolic Execution of Specifications: User Interface and Scenarios
Ralf Kneuper

This report discusses the basic design of the user interface for a system that symbolically executes specifications in order to check them against informal requirements, as a means of increasing confidence in the correctness of the specification. Leading up to that, we give some scenarios which show how symbolic execution can be used for this purpose. This helps to identify the requirements which we have to put on such a system.

The symbolic execution system we describe is intended to be part of the IPSE 2.5; it is to be used as a tool to validate a (formal) requirements, and thus to support the first step in formal software development. It should help the user to analyse and understand a specification before it is implemented.

Technical Report

UMCS-87-12-5 A semantics driven temporal verification system
Graham Gough and Howard Barringer

We present an overview of a SMG, a generic state machine generator, which interfaces to various temporal logic model checkers and provides a practical generic temporal verification system. SMG transforms programs written in user-definable languages to suitable finite state models, thus enabling fast verification of temporal properties of the input program. It can be applied, in particular, to the verification of temporal properties of concurrent and reactive systems.

Technical Report

UMCS-87-12-4 Temporal Logics for Abstract Semantics
Michael D. Fisher

The aim of this thesis is to study the use of temporal logics in the description of abstract semantics for simple programming languages. Following a discussion of the evolution of temporal logics, a coherent framework is described for temporal logics in general. The concept of abstractness and its relationship to compositionality is formalised before considering the use of several different forms of temporal logic in the description of abstract temporal semantics. Though some temporal logics do not easily support the production of such semantics, methods will be described which attempt to enhance these logics sufficiently to enable abstract semantics to be given. It is also shown that semantics given using simple discrete temporal logics do not capture the particular form of abstractness studied here. This study resulted in the formulation of a new class of temporal logics, which, though being a natural extension of other basic temporal logics, also provides a radically different view of the modelling of temporal events.

Ph.D. Thesis

UMCS-87-12-3 Monitoring Tools for Parallel Systems
Heidi W.H. Tang

Monitoring is essential to assess the performance of a computer system. Parallel machines are fundamentally different from conventional machines, and may contain a large number of concurrent activities. Due to their inherently complex nature, a significant amount of experimentation is necessary before they can be built in a cost-effective and viable manner. Therefore new tools and techniques are needed to evaluate their performance, and to aid in their development and analysis.

This thesis describes monitoring computer systems in general, and presents three experimental parallel machine projects currently under research in the University of Manchester: the ParSiFal, Flagship and Dataflow projects.

Two monitoring tools have been designed and implemented in the context of the Manchester Dataflow system, both tools present monitoring data as a graphical, animated display. A program level tool displays a dataflow program executing on a parallel machine. Although the tools have only been used on the Manchester Dataflow system, they have been designed to be as source-machine independent as possible, and they should have wider applicability.

M.Sc. Thesis

UMCS-87-12-2 Performance Evaluation of a Heterogeneous Multi-Ring Dataflow Machine
Y.M. Teo

Advances in semiconductor device technology have made possible the construction of parallel computers, using multiple cheap and powerful processors. Most parallel computers achieve higher performance while retaining the von Neumann model of computation plus associated programming techniques and computer architectures.

Further increases in processor speed, using more exotic device technologies, are beginning to be limited by physical constraints, such as the finite speed of light. The so-called "von Neumann bottleneck" also makes exploitation of inherent program parallelism difficult. To overcome this, several new models of computation and associated machine architectures have been proposed. In particular, the dataflow model of computation, which has no centralised control, has emerged. In this model, instructions are executed when their data is available, and the order of execution is determined solely by the data dependencies among instructions.

A heterogeneous dataflow machine, consisting of a processing element ring and two structure store rings, has been built at the University of Manchester.

This thesis describes a performance evaluation study of the proposed Heterogeneous Multi-Ring Manchester Dataflow Machine, consisting of multiple processing element rings, multiple structure store rings and a global allocator ring.

Firstly, a bottleneck analysis identifies the module constituting the most critical bottleneck in the processing element ring. Architectural changes to remove the bottleneck are proposed. Secondly, a message-based scheme, incorporating a mechanism for the measuring system activity, is proposed for implementing a dynamic, process-based, hardware throttle. The scheme reduces memory requirements in the PE rings by matching program parallelism with available machine parallelism dynamically at run-time. In conclusion, as assessment of the project is made, and some issues that have not been addressed and which require further investigation are discussed.

M.Sc. Thesis

UMCS-87-11-4 Decidability in Temporal Presburger Arithmetic
Wolodymyr Hussak

Decision methods for propositional temporal logics abound. The same cannot be said for first order temporal logics, because most are inherently undecidable. Various reasons account for this, not least of which is the undecidability of the underlying theory to which the temporal operators are added. For this reason, if decision methods are sought, it might appear wise to choose a decidable first order theory at the outset, and only then append temporal operators to the language. However, this still does not guarantee the existence of a decision method, and it may be necessary to subset the full theory.

This dissertation contains the identification of a non-trivial subtheory of a temporal theory of Presburger Arithmetic, possessing a decision method. The decision method offered, comprises two procedures, the second of which is to be executed if the first proves to be inconclusive. The first is a satisfiability procedure based on the algorithm for (non-temporal) Presburger Arithmetic and the second is a tree construction which performs the task required of it, as a result of Konig's Lemma. The subtheory, itself, is quite extensive, and an example is given of how additional formulae, with a greater complexity of temporal description, can be encoded into it.

M.Sc. Thesis

UMCS-87-11-3 An Approach to General Silicon Compilation
Martyn A. Spink

In the present climate, technological advances continue to provide the designers of Computer Aided Computer Design (CACD) systems with a number of problems. More specifically, a CACD system developed to tackle a specific problem is soon outdated by an advance in technology. Such problems particularly apply to systems developed to aid the design of silicon chips, where technological progress is currently especially rapid. Consequently, there is a need to develop a CACD system capable of being adapted to meet the needs of new technological developments. Within such a CACD system, there is a requirement for a silicon compiler general enough to handle not only current design technologies, but also those developed in the future.

The research reported in this thesis has involved an investigation into the requirements for a silicon compiler general enough to handle current and future technological problems. Five criteria, which were used as the basis for the research, are initially defined. These criteria are used as the basis for a discussion of the approaches adopted by other systems to tackle the problems associated with each criterion. Subsequently, the problems that require solution are identified and discussed. In order to solve some of these problems, the developed silicon compiler requires the use of a specifically developed environment. The use of this environment to tackle some of the problems is described, in order to provide an insight into the relationship between the compiler and the environment. Finally, the implementation aspects of two systems are discussed. These systems are used to generate the necessary information required to enable compilation to be performed.

Ph.D. Thesis

UMCS-87-11-2 Manchester Dataflow Machine: Preliminary Benchmark Test Evaluation
John Foley

The Manchester Dataflow Hardware is supported by a Software compiler for the SISAL language and a number of programs have been written to act as Benchmark tests for the hardware. The Benchmark set used contains a wide range of programs including numerical algorithms, sorting, graph colouring and nqueens algorithms plus others. All programs are compiled using a range of optimisations, including function inlining and vectorisation. The resulting statistics, obtained both by simulation and hardware are presented.

Technical Report

UMCS-87-11-1 Formal Specification and Verification of Hardware: A Comparative Case Study
Victoria Stavridou, Howard Barringer and Doug Edwards

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. This article briefly outlines the goals and the philosophy of the HArdware VErification project at the Department of Computer Science within the University of Manchester. Our objective here is to report on the results of a first controlled experiment comparing formalisms and systems that are currently used for formally specifying and verifying both hardware and software systems. Our strategy consists of working with incrementally "harder" test cases, which are used to investigate the characteristics and thus the pros and cons of each formalism. The example used is a purely combinational device.

Technical Report

UMCS-87-8-2 Review of Existing Theorem Provers
Peter A. Lindsay, Richard C. Moore and Brian Ritchie

The report is the result of an extensive study of mechanical support for Formal Reasoning, carried out as part of the IPSE 2.5 project. (It is thus primarily concerned with possible software engineering applications.) An introduction outlines some of the broader issues of FR support: choice of an appropriate logic; grouping of results into "theories"; automation of (certain) deductions; user interaction; heuristics and tactics; proof management in interactive systems. The systems are described individually – including their intended domain of support, their overall structure and important features – with emphasis on how (if at all) they tackle the above issues. Reviewers' impressions are given, together with pointers to relevant literature. The main systems reviewed are LCF, NuPRL, Veritas, Isabelle, IPE, Affirm, Boyer-Moore and Gypsy; there are also brief reviews of other systems. (Resolution theorem provers are not considered.)

Technical Report

UMCS-87-8-1 Throttle Mechanisms for the Manchester Dataflow Machine
Carlos. A. Ruggiero

Dataflow architectures have been studied for several years as an alternative to the von Neumann model of computation. They have proved to be effective in exposing and exploiting different forms of parallelism in programs. However, it has been noted that the parallelism exposed for some programs is much more than the parallelism that can be exploited by specific hardware configurations. This leads to both excessive and unnecessary use of resources (particularly storage).

In this thesis, different software methods for reducing store usage in the Manchester Dataflow Machine by controlling parallelism are presented. Assessment of these methods is achieved by means of simulation. A process- based method is found to be the most suitable and its advantages and problems are discussed in some detail. Among the problems, the necessity of detecting termination of processes is crucial.

Three methods for detecting process termination are analysed and an instruction reference counting method is described in detail. The way this method detects termination for various programming constructs is explained, and the overhead introduced by the method is assessed.

The thesis concludes that there are no fundamental reasons for store usage in dataflow machines to be orders of magnitude higher than in von Neumann machines.

Ph.D. Thesis

UMCS-87-6-1 Specification and Control of Execution of Nondeterministic Dataflow Programs
Pedro M. C. C. Barahona

Dataflow machines are general purpose parallel machines that can exploit various sources of parallelism to speed up program execution. Nondeterministic problem-solving applications can be speeded up if several paths of their state space are searched in parallel. However, the breadth- first search, favoured by the pure dataflow datadriven control of execution, leads to an "explosion" of parallelism with correspondingly high demand on machine resources. Nontrivial problems require a best-first search, guided by a heuristic through the most promising states.

This thesis proposes some extensions to the existing Manchester Dataflow System in order to combine heuristics and exploitation of parallelism for efficient execution of nondeterministic applications, and investigates the suitability of such extended system to support logic programming.

A nondeterministic set data type and a priority construct, introduced in the dataflow language SISAL, enable a flexible control of execution. The set data type prevents "spurious" data dependencies from being introduced in the logical component of the program specification. The priority construct allows the problem-dependent heuristic to be specified separately. The heuristic is taken into account by a process-based control of execution, superimposed on the purely data-driven control.

Simulation of the extended dataflow system has produced encouraging results, which are analysed in some detail. Finally, the main conclusions of the project are summarised together with related topics requiring further investigation.

Ph.D. Thesis

UMCS-87-5-3 Behavioural Implementation Concepts for Nondeterministic Data Types
Tobias Nipkow

This thesis investigates model-theoretic characterizations of behavioural implementation concepts for data types with nondeterministic operations. The main idea of what constitutes a correct implementation is reduced from the level of data types to the level of programs. This provides the basic correctness notion.

The main part of the thesis is devoted to the study of model-theoretic relationships between data types which are necessary and sufficient conditions for correct implementations in the above sense. These model- theoretic characterizations are the basis for correctness proofs of data type implementations.

The two implementation notions studied are loose and robust correctness. The corresponding model-theoretic relationships between data types are simulations and weak simulations. Their soundness and completeness is investigated with respect to two simple stream-based applicative programming languages of different expressiveness.

Ph.D. Thesis

UMCS-87-5-2 How to Choose the Weights in the Knuth Bendix Ordering
Ursula Martin

Knuth and Bendix proposed a very versatile technique for ordering terms, based upon assigning weights to operators and then to terms by adding up the weights of the operators they contain. Our purpose in this paper is twofold. First we give some examples to indicate the flexibility of the method. Then we give a simple and practical algorithm, based on the simplex algorithm, for determining whether or not a set of rules can be ordered by a Knuth Bendix ordering.

Technical Report

UMCS-87-5-1 Some Aspects of Induction
William P. R. Mitchell

Let F be a set of function symbols and R be a rewrite system over $t(F\ U\ n)$ the universal algebra constructed from F and a countably infinite set of variables n. How can we decide whether an equation $M = N$ is an inductive theorem of R or not? Techniques investigated here come under the general notion of 'inductionless induction'. These methods work by completing $R\ U\ \{M = N\}$ in such a way that whilst completing they simultaneously check to see if $M = N$ is an inductive theorem of R.

Technical Report

UMCS-87-4-1 Centrenet Terminal Multiplexer
Evan J. D. Price and Trevor P. Hopkins

This report contains a description of the Centrenet Terminal Multiplexer. This is a microprocessor controlled device that enables terminal devices to be attached to a Centrenet Local Area Network. Currently, a variety of terminals and computers are used in this way, enabling the computer lines to be shared among the terminals. The design of the hardware and a description of the software which controls the Multiplexer is presented.

Technical Report

UMCS-87-2-2 A Hardware Simulator for a Multi-Ring Dataflow Machine
John F. Foley

dataflow machine at Manchester has demonstrated a near linear speedup in program execution when multiple processors are used, provided only that sufficient parallelism is available within the problem. However, the machine has a limited memory bandwidth and this places a ceiling on computation rates; the matching and node stores are particular bottlenecks. One solution is a multi-ring machines consisting of a number of dataflow machines (each with several processors) connected by a switching network and operating in parallel. In order to test such a system, a hardware simulator has been built. This thesis gives details of the machine and the results of initial tests.

Ph.D. Thesis

UMCS-87-2-1 The Parallel Reduction of Lambda Calculus Expression
Paul Watson

Models of computation for the evaluation of Functional Programs are based on the rules for reducing Lambda Calculus expressions. Reduction is achieved by the application of the b-conversion rule to suitable redexes, but few models of computation provide a full implementation of this rule because of the complexity of avoiding variable name clashes. Consequently, evaluation orders are restricted to those in which name clashes cannot occur.

This thesis develops a model of computation for the parallel reduction of Lambda Calculus expressions, represented in De Bruijn's name-free notation, which does provide a full implementation of b-conversion, allowing expression to be reduced by any evaluation order. The model is designed to allow reduction to be performed on a parallel machine comprised of a set of processor/store pairs connected by a communications network. A data- driven, graph reduction execution mechanism is used to exploit the parallel hardware efficiently.

A language for specifying graph reduction models of computation is proposed, and is used to give full specification of the Lambda Calculus reduction model. Specifications in the language can be compiled to the instruction set of a virtual machine. The code produced can then be executed by a virtual machine emulator, or could be recompiled to the order code of a physical processor to allow the high performance simulation of models of computation. The virtual machine is used as the foundation for the design of a physical machine which would support the parallel reduction of lambda calculus expressions.

One of the major issues in the design of graph reduction models of computation is the removal of redundant parts of the expression graph. Neither of the two standard Garbage Collection schemes: Reference Count, or Mark-Scan is entirely suitable for a distributed machine, mainly because of the synchronisation they require. A modified Reference Garbage Collection scheme is described which removes the need for synchronisation, and enhances some of the attractive properties of Reference Count Garbage Collection.

Ph.D. Thesis

UMCS-87-1-2 A Formal Description Method for User Interfaces
Lynn S. Marshall

Formal description methods and user interface design are both important research topics in computer science. This dissertation examines the key issues within these two fields. A formal description method suitable for describing user interface is sought. Flow of control is of major importance in an interactive system. A method suitable for the description of the interface of such a system must clearly illustrate this temporal ordering; it should separate the system into levels of varying detail, and be presented in a modular fashion. The description method should also be concise and understandable while remaining both abstract and mathematically sound. With these criteria in mind, existing formal description techniques are examined. No single existing method is found to be entirely suitable. The finite-state machine, or transition-state diagram (TSD), approach is best for illustrating the flow of control, while a set-theoretic approach, such as the Vienna Development Method (VDM), meets the remaining criteria. Thus a hybrid method is proposed for formally describing interactive systems with emphasis on the user interface. This method uses statecharts, an extended form of TSD, to show the flow of control between operations of the system, along with the pre- and post- conditions of VDM to describe the operations themselves. Examples using the suggested method are presented along with formal semantics and sample proofs.

Ph.D. Thesis

UMCS-87-1-1 An Investigation of the Argonne Model of Or-Parallel Prolog
Kish Shen

The Argonne Or-Parallel Prolog is an all-solution and annotated form of or-parallel Prolog. It may be regarded as a simple extension of sequential Prolog into a parallel environment. Or-parallelism is only exploited where it is to occur and so there are none of the synchronisation and deadlock problems associated with many other approaches to parallelism.

The aim of this project is to study the Argonne model to see if there exists sufficient or-parallelism in normal Prolog programs, and also where the or-parallel annotations should be placed in a program. A simulator has been written in Prolog for this purpose. This simulator allows programs to be run with different numbers of processors, and execution, and the number and types of variable references occurs. The distribution of the reference type directly affects the performance of the model and so is an important statistic.

This report gives an abstract description of the simulator and presents and discusses some results. It also gives an overview of the current state of research on parallel logic programming languages and machines.

M.Sc. Thesis

UMCS-86-11-5 Load Balancing, Locality and Parallelism Control in Fine-Grain Parallel Machines
John Sargeant

A number of important operational issues have been identified by researchers working on many different machine architectures. Load balancing is necessary to distribute work among the processors. Locality is desirable to limit the communication bandwidth which a parallel machine requires. In addition it is necessary to control parallelism to limit resource usage. This document surveys various approaches which have been taken in these areas and proposes solutions in the context of the Flagship rewrite-rule machine project.

Technical Report

UMCS-86-11-4 Design Rule Representation within a Hardware Design System
Julio S. Aude

This thesis examines the problem achieving technology adaptability in CAGD tools.

Design rules currently supported in the data base are concerned with different aspects of design such as timing, loading, signal transmission, layout and power supply distribution and decoupling. These rules are described using a special purpose language, DRDL. Both design and the implementation of DRDL are discussed in this thesis.

In the design rule data base, technologies are organized hierarchically and, in principle, the design rules assigned to a particular technology may be inherited by all its descendants. Selective access to rules by application programs is achieved using attributes to qualify rules and hence control their focus of attention. The internal organization of the design rule data base and the facilities provided to access and update design rule information by the data base management system are also described in this thesis.

Technical Report

UMCS-86-11-2 Unification in Boolean Rings and Unquantified Formulae of the First Order Predicate Calculus
Ursula Martin

Unification, or the solution of equations, in particular algebraic theories has attracted a great deal of attention among computer scientists in recent years, as it is a basic inference mechanism in algebraic manipulation of formulae, automated reasoning and some programming languages. This paper is concerned with unifying Boolean terms, such as those arising in set theory, and the propositional and predicate calculus.

Technical Report

UMCS-86-11-1 Program Logics – A Short Survey
Michael Fisher and Howard Barringer

During the past decade, logics of programs have come to the forefront of computing science. In particular, various programming logics have been shown most useful tools for use in specification, development and verification of complex systems. Although there exist several excellent survey papers covering certain subclasses of program logics, we felt there was a requirement for a broader-based survey that detailed, concisely, the differences between the most common program logics. This report is our first attempt at such a survey.

Technical Report

UMCS-86-10-5 Program Specification and Verification in VDM
Cliff B. Jones

Formal methods employ mathematical notation in writing specifications and use mathematical reasoning in justifying designs with respect to such specifications. One avenue of formal methods research is known as the Vienna Development Method. VDM has been used on programming language and non-language applications. In this paper, programming languages and their compilers are ignored; the focus is on the specification and verification of programs.

VDM emphasizes the model-oriented approach to the specification of date. The reification of abstract objects to representations gives rise to proof obligations; one such set which has wide applicability assumes an increase in implementation bias during the design process. The incompleteness of this approach and an alternative set of rules are discussed.

The decision to show the input/output relation by post-conditions which support decomposition were poorly worked out; those presented below are as convenient to use as the original "Hoare-logic". Recent work on a logic (which is tailored to partial functions) is also described.

Technical Report

UMCS-86-10-2 A Study of an Extended Temporal Logic and a Temporal Fixed Point Calculus
Behnam Banieqbal and Howard Barringer

Propositional temporal logic has previously been extended with right linear grammars to form a family of languages, known as ETL (extended temporal logic), expressively equivalent to w-regular languages. In this report we first show that a set of axioms and inference rules thought to be complete for ETL is in fact incomplete and then provide a complete axiomatisation. Secondly, we introduce an alternative and more natural fixed point extension to propositional temporal logic, for which we provide a decision procedure and establish equivalence with ETL.

Technical Report

UMCS-86-10-1 Using Temporal Logic in the Compositional Specification of Concurrent Systems
Howard Barringer

This article considers the application of temporal logics in the formal specification and development of complex computing systems. In particular, the relevance of compositional proof theories, modularity and abstractness are motivated. A basic technique for obtaining compositional program proof theories, using a temporal language as the assertion language, is demonstrated. Several specialisations for various real parallel programming language frameworks are indicated. Finally, a problem in obtaining abstract semantic descriptions of programming languages in the temporal framework is discussed, together with one particular solution suggesting the use of a logic based on a linear dense time model.

Technical Report

UMCS-86-8-1 FFT on a New Parallel Vector Processor
Kung-Kiu Lau and Xiang-Zhen Qiao

A new parallel processing system has been proposed, and a prototype model of the system has been constructed. It is designed to perform parallel vector operations at maximum efficiency. In addition, it can also handle communicating vector operations, and hence exploit irregular parallelism present in many apparently sequential algorithms. The system is therefore suitable for a wide range of algorithms with varying degrees of parallelism. In this paper, we give a brief description of the system, and discuss the implementation of the Cooley-Tukey FFT on this system. We show that the system's versatility allows it to achieve a near maximum degree of parallelism for this algorithm in the asymptotic case.

Technical Report

UMCS-86-7-1 A Logic for Partial Functions
Jen Huan Cheng

In the formal specification and verification of computer software, a logic plays a vital role. The classical logic is inadequate because it only handles total functions, whilst partial functions are indispensable in programming.

Criteria for a specification logic that covers partial functions are discussed, and against these existing methods and related logics are surveyed.

A first-order logic for partial functions (LPF) is formalized. This is achieved in several stages; propositional calculus, predicate calculus and predicate calculus with equality. In each stage, various meta-theorems such as the consistency and the completeness are established. Different methods are adopted for different stages: Hilbert-style for the propositional calculus, the sequent calculus for the predicate calculus and an algorithm of congruence closure for equality axioms. Other established properties are: the expressive completeness of the propositional connectives and the cut- elimination theorem. An extension of LPF to the second-order is proposed as well.

To make LPF easily usable by humans, a different proof style, namely the natural deduction, is considered, and its relation to the sequent calculus is set up. In order to achieve a practical comparison with other logics and an automated LPF supporting formal methods of software development, a proof checker that allows users to define their own logics is proposed and specified.

Ph.D. Thesis

UMCS-86-5-1 Multiset Orderings
Ursula Martin

A multiset or bag on a set S is an unordered sequence of elements of S. This paper is concerned with the construction on M(S), the set of multisets on S. Such orderings have been studied for many years in invariant theory and the theory of partitions, and also more recently have proved particularly useful in proofs of program termination and in constructing orderings which prove the termination of term rewriting systems. In this paper we shall describe a geometrical approach to the study of orderings on multisets. We shall show that any ordering on M(S) satisfying certain natural restrictions (a "tame" ordering) arises from a particular subset of R called a cone, and conversely any cone gives rise to a tame ordering on M(S). Tp demonstrate the power of this geometric approach we shall classify all tame orderings on R , and describe an interesting new multiset ordering which inherits a partial order on S. We shall show how the dominance ordering, which has been much studied when S is totally ordered, can be extended to the case when S is only partially ordered, and describe how the standard multiset ordering can be interpreted geometrically. Finally we shall show that any tame total ordering on R is related to the lexicographic order on R by the action of an invertible matrix.

Technical Report

UMCS-86-4-4 Approaches to Raster Graphics and Image Manipulation
Mark van Harmelen

Several approaches to raster graphics and image manipulation are explored in this thesis, with the end aim of creating an expandable and flexible generic architecture with different capabilities and performances. The task is approached using two routes which are: (i) An examination of the nature of innovation in computer architecture, with particular emphasis on factors which enhance the performance and capabilities of the systems. In the computer graphics and image manipulation domain this indicates that attention should be paid to parallelism, and to VLSI and videodisc technology. (ii) An examination of abstract process-data models for computer graphics and image manipulation, to see how parallelism can be exploited in that area.

A generic architectures is then produced for a satelite computer graphics and image manipulation system. The generic architecture may be instantiated in different ways, producing machine architectures with a variety of different capabilities and performances. The generic architecture contains a bus structure that enables the exploitation of non-deterministic parallelism for graphics tasks, and separates out the high level control information from the low-level pixel data. Thus multiple processors may be used for performance enhancements in certain tasks, and different kinds of proccessors may be incorporated into a particular instantiation to create machines with different capabilities.

In particular, the architectural framework uses a Rotating Bus as the primary means of communication between multiple processors. A dedicated subsidiary bus is used for transfer of pixel information between pixel manipulation processors, frame buffers and video disc interfaces.

The thesis contains sufficient low-level hardware detail to form an informal proof as to the feasibility of the approach outlined above, concentrating on the mroe critical areas of the hardware; bus and frame buffer design.

Ph.D. Thesis

UMCS-86-4-3 Stored Data Structures on the Manchester Dataflow Machine
John Sargeant and Chris C. Kirkham

Experience with the Manchester Dataflow Machine has highlighted the importance of efficient handling of stored data structures in a practical parallel machine. It has proved necessary to add a special-purpose structure store to the machine, and this paper describes the role of this structure store and the software which uses it. Some key issues in data structure handling for parallel machines are discussed.

Technical Report

UMCS-86-4-1 An Implementation of the GKS-3/PHIGS Viewing Pipeline
Karen Singleton

Following the establishment of GKS as the first international standard for computer graphics, two new standards, GKS-3D and PHIGS, are emerging which cater for three dimensional graphics. In order to display a 3-D object, a description of how it is to be viewed must be given. GKS-3D and PHIGS provide a viewing pipeline to generate the parallel or perspective view required. This paper describes in detail the functionality and an implementation of the GKD-3D/PHIGS viewing pipeline.

Technical Report

UMCS-86-3-1 Centrenet Network Monitor
Trevor P. Hopkins

This report contains a description of the Centrenet Network Monitor system. This device monitors traffic on a selected mode in the network, and permits the user to display some (or all) of the packets routed through that mode in a variety of formats. The system is transparent to network users, and requires no modification to hardware or software. The Network Monitor is primarily intended as a test and debugging tool for use with protocol development.

Technical Report

UMCS-85-10-3 Processor Allocation in a Multi-ring Dataflow Machine
Pedro Barahona and John R. Gurd

Partitioning a program into processes and allocating these processes to different processors determines the efficiency of a multiprocessor architecture. In the dataflow model, where each process consists of a single instruction (e.g. an arithmetic or logic operation), an automatic decomposition of a program into fine-grain processes can be achieved. A prototype dataflow machine is in operation at the University of Manchester. The machine comprises a single processing element (PE), which contains several units connected together in a pipelined ring. A Multi-ring Dataflow Machine (MDM), containing several such processing elements connected together via a switching network, is currently under investigation.

This paper describes a method of allocating the dataflow instructions to the processing elements of the Multi-ring Dataflow Machine and examines the influence of the method on the selection of a switching network. Results obtained from simulation of the machine are presented, and it is shown that programs are executed efficiently when their parallelism matches the parallelism of the machine hardware.

Technical Report

UMCS-85-10-2 Efficient Dataflow Code Generation for SISAL
Anton P. W. Bohm and John Sargeant

The efficiency of dataflow code generated from a high level language can be improved dramatically by both conventional and dataflow-specific optimisations. Such techniques and used in implementing the functional language SISAL on the Manchester Dataflow Machine.

Technical Report

UMCS-85-10-1 Non-deterministic Data Types: Models and Implementations
Tobias Nipkow

The model theoretic basis for (abstract) data types is generalized from algebras to multi-algebras in order to cope with non-deterministic operations. A programming oriented definition and a model theoretic criterion (called simulation) for implementation of data types are given. To justify the criterion with respect to the definition, an abstract framework linking denotational semantics of programming languages and model theory of data types is set up. A set of constraints on a programming language semantics are derived which guarantee that simulation implies implementation. It is argued that any language supporting data abstraction does fulfill these constraints. As an example a simple but expressive language L is defined and it is formally proved that L does conform to these restrictions.

Technical Report

UMCS-85-9-3 Up and down the Temporal Way
Howard Barringer

A formal specification of a multiple lift system is constructed. The example illustrates and justifies one of many possible system specification styles based on temporal techniques.

Technical Report

UMCS-85-9-2 Image Transfer by Packet-switched Network
Trevor P. Hopkins

The advantages and disadvantages of using packet-switching technology for the transfer of image information in real time are considered. An experimental implementation of parts of a system based on a high-speed Local Area Network is described; these include a simple scren output device and a real-time camera input device. The generation of images using a number of microprocessors is also described. A number of applications for such a system are investigated and the extension of this approach to implement in Integrated Information Presentation system is considered.

Technical Report

UMCS-85-9-1 Extending Pascal with One-entry/Multi-exit Procedures
Ian D. Cottam

To make the verification of Pascal programs more manageable their so-called standard and exceptional behaviour is separated syntactically with a simple exception handling mechanism (due to Flaviu Cristian). The programming language Pascal (BS 6192 / ISO 7185 level 1) is augmented with the guarded statement sequence that may signal a (parameterised) exception, and exception handlers.

Two different implementations have been undertaken; a Pacal compiler has been modified to accept and compile the extensions, and a stand-alone preprocessor from the extended language to standard Pascal has been developed. The merits, and otherwise, of both strategies are reported. The constructs are presented informally; a companion paper will give the formal semantics.

Technical Report

UMCS-85-8-2 Efficient Stored Data Structures for Dataflow Computing
John Sargeant

Dataflow computers can exploit many different forms of parallelism in programs. Dataflow languages are high-level functional languages which allow many sorts of parallel computation to be expressed naturally. Dataflow machines executing programs written in such languages potentially offer very high computing speeds for a wide range of applications. One of the main problems in dataflow computing to date has been the handling of data structures. The concept of stored data structures must be integrated into the dataflow model of computation at reasonable cost.

This thesis describes how this can be done, particularly for the Manchester Dataflow Machine. It is necessary to tackle the problems on several levels, including the underlying hardware. An extension to the architecture, a structure store, is described. The code generated for the dataflow language SISAL is discussed in detail. Particular attention is paid to the issue of garbage collection. A number of optimisations which can dramatically improve the quality of dataflow code are described. As a result of this work, it is now possible to generate from SISAL dataflow code which is efficient in terms of the total number of instructions executed, while retaining the ability to exploit large amounts of parallelism. Simulation results which demonstrate this are presented.

Ph.D. Thesis

UMCS-85-8-1 The Unification of Terms: A Category-Theoretic Algorithm
D. E. Rydeheard and R. M. Burstall

As an illustration of the role of abstract mathematics in program design, an algorithm for the unification of terms is derived from constructions of colimits in category theory.

Technical Report

UMCS-84-10-2 Performance Evaluation of a Multi-ring Dataflow Machine
Pedro Barahona

The integration attained by VLSI technolgoy has made it possible to build parallel computers composed of several cheap and powerful processors. Early parallel architectures offered performance improvements whilst retaining the underlying von Neumann model of computation. The von Neumann model makes the exploitation of inherent parallelism of programs problematical. To overcome these problems, other models of computation, and machines to implement them have been proposed. In particular, the dataflow model has no centralised control and no spurious sequencing of instructions. Instructions are executed as soon as their input data is available.

A prototype dataflow machine has been built at the University of Manchester, composed of several units pipelined together in a ring. This thesis is a preliminary attempt to assess the performance of a Multi-ring Dataflow Machine built with several processing elements, each similar to the prototype ring implemented.

The Switch required by such a Multi-ring Dataflow Machine and the allocation of the dataflow instrucitons to the processing elements are discussed in detail. Simulation has been used to predict the machine performance. Minor changes to the ring architecture are proposed in order to avoid inefficiencies that were discovered. The main conclusions of the project are presented together with topics requiring further work.

Technical Report

UMCS-84-10-1 A Database Programming Language: Definition, Implementation and Correctness Proofs
Ann Welsh

A high level binary relational database programming language is designed using formal specification. The restrictions imposed by the underlying data model are minimised, thus removing the distinction between implicit and explicit database constraints and increasing the freedom of the user. A new method is devised for handling semantic integrity by means of program correctness proofs, rather than the conventional approach of dynamic checking. For this purpose, a set of database-oriented Hoare-style axioms and proof rules are designed for a subset of the language. This method avoids many associated constraint handling problems such as efficiency of implementation and action on constraint violation. The implementation of the language on MDB, a formally specified database system, is described.

Technical Report

UMCS-84-2-1 The Formal Semantics of Deterministic Dataflow Programs
Jose N. F. Oliveira

For more than a decade, research into dataflow computing has been targetted at designing novel architectures which attempt to supersede the von Neumann design for efficient, parallel execution of computations. Research at Manchester University has led to the implementation of a prototype dataflow computer which has been operational since 1981. During the same period, a dramatic reduction in hardware costs has uncovered deficiencies in software development technology, which has been unable to produce the required quantity of cheap, reliable and easily maintainable software.

This thesis is an attempt to assess the impact of dataflow computing on software technology. The Manchester dataflow (MDF) system design is used as an example for this study. Semantics for the MDF-machine are desribed at the usual graphical level using notation characteristic of dataflow programming. Properties of the MDF-system are studied from both theoretical and practical points of view. Based on this semantics, methods for constructive design and proof of programs written for the MDF-system are proposed. A transformational approach to dataflow program development based on functional programming is presented as the basis for a more ambitious reasoning methodology.

Ph.D. Thesis

UMCS-83-10-1 A Denotational Semantics for Concurrent Ada Programs
Ian Mearns

The dynamic semantics for a subset of the Ada tasking constructs is presented. The constructs denote processes, following the general theory developed by de Bakker and Zucker, and do not require translation into an intermediate language. All operations on processes are designed and theoretically justified. An existing proof system for Ada tasks is revised, and proved to be sound and relatively complete with respect to the semantics.

Technical Report

UMCS-82-10-1 The Specification, Design and Implementation of NDB
Ann Welsh

An attempt is made to become familiar with the "rigorous" method of software development and to identify some associated problems, using the binary-relational non-programmer database system, NDB, as a vehicle for this research.

A complete specification and development of NDB, and an implementation in Pascal, are produced.

The decomposition of specifications and designs into manageable and resuable units is seen to be desirable, especially when concerned with a large and reusable system.

Technical Report

UMCS-80-3-1 A Multilayered Data Flow Computer Architecture
John Gurd, Ian Watson and John Glauert

This paper introduces a multilayered data flow computer architecture based upon a high level language and underlying graphical notation for representing computations.

The language, LAPSE, employs a single assignment rule and several more familiar parallel programming constructs. It permits both iteration and recursion.

The graphical notation utilises the concept of labels for those tokens which use computational subgraphs reentrantly. The operation of labels in implementing iteration, recursion and data structures is illustrated. The architecture is based on a ring-structure in which binary representations of tokens, known as results, circulate. A processing unit computes results as required by a stored program which represents the computational graph. Each result carries with it a name which is derived from the notational label. Names perform two functions : Firstly they separate different instantiations of reentrant code; secondly they can be associatively matched with other inputs to their destination node, to determine when the node may be executed. The ring-structure can be pipelined to achieve an instruction execution rate which is limited by the amount of parallel activity permitted by the program and the technology used.

An extensible version of the architecture contains many such rings in a multilayered structure whose execution rate is bound solely by the inherent parallelism of the programs running on it. The architecture can also be organised so as to exhibit a high degree of tolerance to hardware component failures.

Technical Report