TECHNICAL REPORT NO. 171

INDIANA UNIVERSITY

COMPUTER SCIENCE DEPARTMENTAL REPORT

MAY, 1985

Computer Science
Departmental Report
May, 1985

Table of Contents

FACULTY AND INTERESTS

Daryel Akerlind, Assistant Professor; Ph.D., Australian National University. Theory of computation, computational complexity.

John Barnden, Assistant Professor; Ph.D., Oxford. Artificial intelligence, programming languages.

John Buck, Lecturer; B.S., Virginia Tech. Systems analysis, data base and information systems, computer science education.

James Burns, Assistant Professor; Ph.D., Georgia Tech. Theoretical computer science, parallel and distributed systems, distributed data bases, computer graphics.

Will Clinger, Assistant Professor; Ph.D., Massachusetts Institute of Technology. Semantics of programming languages, artificial intelligence, nondeterministic concurrent computation, logic.

Kent Dybvig, Assistant Professor; Ph.D., University of North Carolina at Chapel Hill. Functional Programming Languages and their implementation.

George Epstein, Professor; Ph.D., University of California, Los Angeles. Systems design, multiple-valued logic, computer science education.

John Franco, Assistant Professor; Ph.D., Rutgers University. Analysis of algorithms, Theoretical computer science, Combinatorial optimization, NP-complete problems.

Daniel Friedman, Professor; Ph.D., University of Texas at Austin. Programming languages.

Dennis Gannon, Associate Professor, Ph.D., University of Illinois. Parallel computation, computer architecture, programming systems and mathematical software.

Stanley Hagstrom, Professor, Computer Science and Chemistry; Ph.D., Iowa State University. Computer hardware, laboratory automation, computer networking, operating systems, software engineering, analysis of algorithms in *ab initio* quantum chemistry.

Christopher T. Haynes, Assistant Professor; Ph.D., University of Iowa. Programming Languages and operating systems.

Steven Johnson, Assistant Professor; Ph.D., Indiana University. Applicative programming, multiprocessing architectures and languages, hardware/software specifications.

Stan Kwasny, Assistant Professor; Ph.D., Ohio State University. Natural language understanding, artificial intelligence, data structures, computational linguistics, data base systems.

Diane Kewley-Port, Assistant Professor; part-time, Ph.D., City University of New York. Voice I/O for human-computer communication, speech recognition.

John O'Donnell, Assistant Professor; Ph.D., University of Iowa. Computer architecture, operating systems, VLSI design, programming languages.

Franklin Prosser, Professor, Ph.D., Pennsylvania State University. Digital hardware, operating systems, computer science education.

Paul Purdom, Professor; Ph.D., California Institute of Technology. Analysis of algorithms, compilers, rewriting systems.

Edward L. Robertson, Professor and Chairman; Ph.D., University of Wisconsin. Theory of computation, computational complexity, hardware and software systems architecture, data bases.

L. David Sabbagh, Associate Professor, part-time; Ph.D., Purdue. Graphics.

Dirk Van Gucht, Assistant Professor, Ph.D., Vanderbilt University. Database theory, graph theoretical applications in computer science, sequential and parallel algorithms, complexity theory, and artificial intelligence.

Mitchell Wand, Professor; Ph.D., Massachusetts Institute of Technology. Semantics of programming languages, logic, algebra.

David Winkel, Professor; Ph.D., Iowa State. Digital Design, Applicative architectures.

David Wise, Associate Professor; Ph.D., University of Wisconsin. Applicative programming, data structures, multiprocessing architectures and languages.

CURRENT FACILITIES

The heart of the research environment at Indiana University Computer Science Department is a VAX 11/780. This unit includes floating point accelerator and eight megabytes of RAM on two interleaved memory controllers. Storage devices include two 28 Mb RK07s on Unibus, three 474 Mb Fujitsu Eagle Winchester drives with a S.I. 9900 controller on Massbus, three 169 Mb Fujitsu Winchester drives with a S.I. 9400 controller on an additional Massbus, one STC 6250 BPI streaming tape drive on a S.I. Massbus Controller, and one Pertek 1600 BPI tape drive on a Wespercorp Unibus Controller. Communications include seven new Interlan NTS-10 ethernet terminal servers, and one new Interlan Unibus ethernet controller. The terminal servers are capable of supporting 64 serial ports and are now serving 40 remote terminals. Users include all Computer Science Department faculty and staff plus a small number of advanced graduate students.

Technical personnel staff a service, repair, and maintenance shop, which is available to researchers. This facility supports in house maintenance for the majority of the Department's equipment. Diagnostic equipment includes a Tektronix DAS 9100 Logic Anaylysis System, Tektronix Logic Analyzer, Tektronix oscilloscope Model 466, Tektronix oscilloscope Model 475, Hewlett Packard oscilloscope Model 1700A, and other assorted diagnostic equipment. On hand are a replacement stock of integrated circuits, transistors, and assorted electronic parts and supplies. The shop also stocks a working quantity of integrated circuitry for research and prototyping uses.

The Data Structures Laboratory for students consists of 16 MC6809-based workstations, 14 of which are Smoke Signal Chieftan Microcomputers, interfaced with Lear-Siegler Adm3a terminals. Two workstations have Decwriter IV printing terminals and Anadex 9500 line printers.

The Senior Laboratory contains 24 Motorola 68K Educational Computer Boards (ECBs), each of which interfaces to an ADDS Viewpoint terminal. The host port on each ECB interfaces through the campus Sytek data network to a configuration of VAX 11/780s, belonging to the Bloomington Academic Computing System. For classroom use, the remote host supports a cross-development environment for the 68000.

The Computer Structures Laboratory includes 17 Logic Engines and 50 backplanes. The Logic Engines, designed by I.U. faculty and produced by Logic Design, Inc., include two 5 1/4" floppy disk drives, a 6809 CPU, and a backplane with flexible and varied methods of constructing differing configurations of computers. During the semester, the class produces an LD16 minicomputer, which is a version of the PDP 8 design. Logic Engines, which greatly facilitate microprogram design and development, play a prominent role in several proposed research projects.

An additional laboratory is equipped with an Intel 432 development system, consisting of an Intellec series III, an 86/330 system, and the 432/600 itself.

The Department's printed circuit fabrication facility consists of manual drafting equipment, a reduction camera, a photography darkroom and enlarger, and a newly-acquired Model 41 Gerber Photoplotter (Model 3100 controller), which will interface to five Hewlett-Packard 9836 CAD workstations, via a Hewlett-Packard 3000/44 serving as a spooler. Each of the 9836 stations is equipped with an Aydin Controls 19 inch color monitor. Other output devices include a HP 7580 series wide-bed color plotter, and a 7221 series flat-bed color plotter. Other P.C. fabrication equipment includes a Scanex, a Riston laminator, two drying ovens, copper etching facilities, two P.C. drills, and supplies.

The Hewlett-Packard 3000/44 has been used to host courses in Informations Systems and Operating Systems.

INSTRUCTIONAL REVIEW

The instructional programs of the Department are based on developing the capabilities of analysis, precision and abstraction, on providing the fundamental concepts and principles underlying computer systems, and on exercising all these in substantial projects designed from fundamental principles and directed toward real-world goals.

Although one cannot imagine a successful programmer unable to work with great complexity, a person whose skills are limited to these is a mere "hacker". To be a true computer scientist, whether in academia or in an applied setting, one must be able to identify recurring patterns of operations and objects, delineate these patterns with boundaries and parameters which are simple yet powerful, and use these developed abstractions to escape the complexities of one level and construct new structures on a higher level. In a simple computer science example, we expect a student in Data Structures not only to write a program which builds and maintains balanced trees but also to use tree balancing in conjunction with other structures and operations.

In stressing fundamental principles of computer science, we are joined by a large number of forward looking computer science departments. Our special emphasis on the deep foundations is exemplified in our Programming Languages course. Unlike most courses with this title, which present a "smorgasbord" of syntax and semantics from numerous programming languages, ours concentrates on one language (Scheme, a dialect of Lisp) which illustrates basic language properties and allows students to build for themselves many of the features of other common languages. Another component of our educational philosophy, the use of projects to integrate skills and knowledge, is reflected in two kinds of courses: the individual project courses, required for the BS and an option for the BA, and the full-year advanced course sequences.

The full-year advanced courses used both as "senior sequences" and options in the MS program, are a feature unique to our department. A full academic year to present material allows for an in-depth presentation of the principles and practicalities of an area. It allows for a well-paced, carefully designed project which requires students to apply the skills of analysis, precision, and abstraction to the knowledge of the specific field. There is no need to choose between projects which are either brute-force because they start so early that students have not seen the fundamental principles or slap-dash because they begin so late that there is no time for careful completion. An operating system project traditionally involves a kernel with memory management and process control plus rudimentary file and other utilities. A group of students in information systems/databases will design and develop a complete system for some application. Systems have been done for other University departments, a local hospital, and local and state governments (a few projects have even resulted in publications). We feel that these projects reinforce the principles and refine the skills of our students, providing a final tempering for our very finely honed computing professionals.

While fundamental knowledge and skills are the building blocks of our instruction, the continuing emphasis on design is the matrix which unites our program. The skills of abstraction are especially related to design and are exercised from the beginning of the curriculum. Even the digital hardware course holds design to be of central importance.

Current Curriculum

The early part of the curriculum includes:

C201: Introduction to Computer Programming
usually PASCAL or FORTRAN

C335: Computer Structures
a look inside the box

C251: Found. of Digital Computing
Discrete Math, Logic

C343: Data Structures

C311:Programming Languages

The full-year advanced sequences provide specialization in:

*information systems
*assemblers and compilers
*operating systems
*computer hardware design and construction
*artificial intelligence
*theory of computation
*numerical analysis

One sequence is required for the B.A. degree, two for the B.S. The B.S. also requires more math and science and an individual programming project. Of particular interest to prospective B.S. majors is the optional minor in Business, which includes junior-level courses for the School of Business B.S. program.

Demographics

	1974	1979	1984
undergrad majors	60	180	658
grad majors	32	79	153
faculty	9	11	19
staff	2	4	8

Each year between 1978 and 1983 the number of students entering Indiana University wishing to major in computer Science grew by 55%. This is amazing both in the magnitude and the consistency of this figure.

Summaries of Research Projects

Wise, David S. and Friedman, Daniel P.

Applicative Programming for Indeterminate Systems

Indeterminate systems are operating systems, data base systems, distributed systems, and multiprocesser systems that require real-time response to many independent conditions that occur with a relative synchronization not known at the time the system is built. Applicative or functional programming is a style of expressing computer algorithms as mathematical function definitions, which are specifically devoid of time- and side-effects.

Extending applicative programming to deal with the essential properties of timing in systems is important because it is already so promising for efficient use of these same systems for time-independent tasks. Traditional programming styles do not make good use of parallelism available in new architectures because they have been modelled after single processor computers. They have however, been extended to deal with synchronization issues. Applicative languages, which can well use parallelism unknown to the programmer, have not been extended to deal with all of the problems in working with such systems.

We propose to develop the DAISY programming languages and DSI system to a production environment that can cope with issues of indeterminism among input conditions, breadth-first evaluation of condition trees, failures of output devices, and embedded systems, and to refine system performance with respect to program maintenance and automatic introduction of sequentiality to reduce time and space needs.

Wise, David S.

Methods and Architectures for Applicative Programming

We wish to advance our study of applicative languages as a basis for general purpose programming. This research has two main aspects: the formulation of constructs and methods required for robust programming endeavors, and the construction of hardware that supports the tenets of programming that we espouse. In previous investigations we have isolated a small number of language constructs that serve to specify a surprisingly broad class of applications, including those that commonly arise in systems programming. We have added "data recursion" and an indeterminate constructor to the applicative vocabulary. With data recursions cyclic behavior can be represented by infinite data structures. The indeterminate constructor is used to address concurrency and real-time behavior. These constructs are related. Each enforces the programmer's view of activity as an attribute of data and thus its treatment in spatial terms.

Our research has focused on the individual programmer, and yields methods that we claim improve productivity. The crucial question is whether these methods work "in the large", for example, in applications involving many participants. To explore such issues we propose to enhance an existing, purely applicative programming language into a programming system.

In its present form the language implements "suspending construction", an operational model of computation that we proposed in the mid 1970s. We believe this model can serve as a basis for a multiprocessing host that supports efficient execution of applicative programs. We propose a sequence of prototype architectures through which we shall explore this model.

Prosser, Franklin

Structured Hardware Design

The theme of my major research interests is to make hardware design processes more systematic, structured, and orderly. This theme has led me down two major avenues: teaching of structured digital hardware design, and the development of design aids to support structured design. In both of these areas I have worked closely with Professor David Winkel.

In our research in structured design we have developed a Logic Engine -- a system for microprogrammed control of hardware architectures. The system has (a) a powerful base unit for clocks, display, and power, (b) a large printed circuit board containing the Logic Engine controller, a microcomputer system for the user's hardware design, and (c) a software support system to assist the user in developing and testing microcode and debugging the overall design.

Using the Logic Engine as the control element, we are developing a SCHEME machine for high-speed execution of programs written in the SCHEME language.

On a more abstract level, I have been working to develop structured design methods and incorporate them into routine hardware design. Among the techniques that we have found most useful are mixed logic for circuit descriptions and the ASM description (after Osborne and Clare) of control algorithms. Recently, I have made some progress in incorporating these and similar techniques into VLSI design.

Robertson, Edward L.
National Science Foundation
7-1-80 6-30-82

Studies Related to NP-Complete Problems, Structure, Approximation and Backtracking

Important practical problems in the class of "NP-complete problems," from such diverse areas as industrial management and computer network reliability, are all difficult to solve, in the sense that all known general methods for solution are not much better than trying all possible cases. Morever, these problems are all related, so that an efficient solution method for any one problem would provide efficient solutions for all of them. This research intends to characterize NP-complete problems and methods for finding approximate solutions to these problems.

Robertson, Edward
National Science Foundation
6-1-83 11-30-84

Computer Science and Computer Engineering Research Equipment

Equipment will be purchased to augment the department's VAX 11/780 system and the digital design lab. The department will also obtain high-resolution graphics workstations and a terminal screen microprocessor/network interface.

Robertson, Edward
National Science Foundation
9-4-84 5-31-85

Realtime Semantics

This project aims at the development of a state machine language to express realtime control systems. The definition of the language may also be interpreted as the specification of a state machine built with hardware components as opposed to relying on a language for its semantics.

We also propose to develop algorithms to translate ladder diagram languages into some high level language to make application and study of ladder diagram languages in terms of state machines effectively manageable.

Franco, John
Air Force Office of Scientific Research
9-30-84 9-29-85

Probabilistic Analysis of Algorithms for NP-Complete Problems

The goal of this research is to develop and analyze algorithms which can, in some practical sense, solve certain NP-hard problems quickly.

The problem we are primarily interested in is the SATISFIABILITY (SAT) problem, and we are also looking at algorithms for GRAPH COLORABILITY and MAXIMUM INDEPENDENT SET.

We have chosen a constant-clause-size distriubtion to model instances of SAT not because it reproduces samples of instances as they occur in practice (it doesn't) but because it leads to results which seem to hold for a variety of samples. Let E be a distribution on instances of SAT defined as follows: an instance contains n clauses each selected uniformly and independently from Q(k,V).

A number of algorithms that have been shown to perform well probabilistically under E. We have shown that the Pure-Literal heuristic can find a solution to a random instance of SAT (under E) in polynomial time with probability approaching 1 when the limit of n/r as n and r approach infinity is less than 1. Also, we have shown that the Unit-Clause rule can be used to find a solution to a random instance of SAT in polynomial time with constant probability (this suggests an algorithm which is efficient in probability) when the limit of n/r is less than

$$2\frac{k-1}{k}\left(\frac{k-1}{k-2}\right)k-2$$

and that a modification of this rule can be used to find a solution to a random instance of SAT efficiently when the limit of n/r is less than approximately

$$3.09\frac{*2}{k+1}k-2\left(\frac{k4}{k-2}\right)k-2$$

Instances generated according to E nearly always have solutions when the limit of n/r is less than approximately 2**(k-1) and nearly always have no solution when the limit of n/r is greater than 2**(k-1).

Barnden, John A.

Information Processing using Diagrammatic Imagery

An unconscious-imagery model of human cognition is under investigation in an artificial intelligence project. In the model, temporary symbolic structures appear as imaginal diagrams analogous to real drawings of data structures.

The diagrams are states of abstract array-like structures implemented as neural networks. The present study is attempting to elucidate the expressive and problem-solving convenience of the unusual (e.g., hybrid pictorial/abstract) symbolisms natural to the model. The project also involves the computer simulation of the necessary image-manipulation processes.

Barnden, John A.

Representation of Beliefs and Other Propositional Attitudes

The representation of beliefs, wants and other "propositional attitudes" is being investigated, for the purposes of artificial-intelligence computer programs. The situations of interest are those in which there is a set of cognitive agents having beliefs, wants, etc. about each other, and, especially, about each other's beliefs, wants, etc. Some intricate representational issues are raised by such situations, and have been an important subject of artificial-intelligence research in recent years.

Epstein, George

Discrete-Valued Logic and Functions

Related to course work in Logic Functions and Machine Theoryu, C525, research is being performed on various kinds of discrete-valued functions. Since 1982, papers have been published on symmetric functions, positive functions, threshold functions, and orthogonal functions. The latter is now the subject of doctoral work by my student, R. R. Loka. Fuzzy functions are now being studied in C525 and with M. Mukaidono, Meiji University of Japan. A textbook on "An Introduction to Foundations of Computer Science" is now under progress.

Purdom, Paul

Term Rewriting Systems

Term rewriting systems replace quantities with equal quantities according to a set of rules, with the goal of obtaining the simplest form of the initial expression. Rewriting is a powerful paradigm with applications to theorem proving, abstract algebra, program verification, expert systems, and compiler design. The Knuth-Bendix procedure is a method for developing efficient rewrite systems. It uses many basic algorithms that are important to the fields of theorem proving and abstract algebra. The purpose of this proposal is to develop methods for large improvements in the efficiency of the Knuth-Bendix procedure, both the individual algorithms and the overall procedure. These improvements will greatly extend the practical range of application of the Knuth-Bendix algorithm. The development of more efficient term rewriting algorithms will make it feasible to solve many problems by writing a simple set of rewrite rules rather than by writing a complex program.

Expert systems are an important application of computers. Such systems involve a large base of knowledge, some rules to make inferences from the knowledge, and a simple inference mechanism. The reason for a simple inference mechanism is speed. The current practice is to compensate for the deficiencies of the inference mechanism by adding additional knowledge and rules. Rewrite rules form a simple and easy to use method to specify the computations that one wants done (without any need to give the details as to how the computations are to be carried out). A speedy implementation of rewriting techniques appears to be an extremely promising tool for implementing sophisticated expert systems.

Purdom, Paul W.

Game Playing

Traditional game playing programs are based on combining limited depth search, heuristic evaluation functions, and minimax propagation of values. Nau has shown that for some games this approach leads to a paradox; deeper searching leads to more random play (which is worse play if one has a reasonable heuristic evaluation function).

We are developing new approaches to game playing that avoid the paradox. So far we have developed a theory for how to use heuristic search to make the best first move (assuming a perfect player will take over and make the rest of the moves for you). We are currently investigating how to make the best move when the program, with all of its imperfections, must play the entire game.

Purdom, Paul W.
National Science Foundation
7-1-83 12-31-85

Analysis of Algorithms for NP Complete Problems

This research has three ultimate goals: 1) the development of improved algorithms for solving NP-complete problems 2) learning more about which problems in NP-complete problems sets can be solved rapidly, & 3) finding random sets of NP-complete problems that are difficult to solve. My approach will be to do average time and worst-case time analyses of algorithms that solve NP-complete problems. I will use the information I obtain about the causes of inefficiency in present algorithms to develop improved algorithms.

Mitchell Wand, Daniel P. Friedman
National Science Foundation
9-1-81 8-31-83

Semantics Issues in Computation

We will continue our studies on the relationship between denotational semantics and implementations. In particular, we hope to extend our studies of compiler correctness to incorporate more realistic machine architectures, including problems of data types and storage management. We hope to extend our use of tractable proof techniques to these problems. We also intend to explore problems of representations in more general settings including those of reflective processors and parallelism and non-determinism.

Daniel P. Friedman, Christopher T. Haynes
National Science Foundation
6-15-83 11-30-85

Constraining Control

Continuations, when available as first-class objects, provide a general control abstraction in programming languages. They liberate the programmer from specific control structures, increasing programming language extensibility. However, the full power of continuations is not required in many applications, in which case it may be desirable to appropriately restrict them in the interest of security and efficiency. These restrictions may be enforced by embedding continuations in functional objects. In this research we explore uses of continuations and techniques for restricting them.

Akerlind, Daryel S.

Provable Conditions in Computational Complexity

My investigations into dynamic complexity measures have shown the existence of severe limitations on what can be demonstrated (proved within a formal axiomatic system) about algorithm performance. These limitations reflect the existence of "pathological" or "anomalous" algorithms --- that is, algorithms for which there is a large discrepancy between their performance and what can be demonstrated about their performance. Such "anomalous" algorithms occur at all levels of dynamic complexity --- even very fast algorithms can be "anomalous".

Continuing research aims to extend these results and seeks criteria, based on size or structural complexity, that will characterize (i) classes of algorithms free from "anomalies", and (ii) classes of algorithms for which the extent of the anomalous behaviour is bounded.

June 5, 1985

Grants — May, 1982 – April, 1985

48 297 03 – NSF MCS 77–22325, David Wise/Daniel Friedman, "Applicative Programming for Systems," $160,514 3/1/78–11/30/82.

48 297 07 – NSF MCS 80–04337, Edward Robertson, "Studies Related to NP-Complete Problems, Structure, Approximation and Backtracking," $66,237 7/1/80–12/31/83.

48 297 09 – NSF SER 80–13219, Frank Prosser, "Improvement of Laboratory in Computer Structures," $19,800 10/15/80–9/30/83.

48 297 10 – NSF MCS 81–02291, Robert Filman, changed to John Barnden/William Clinger, "Meta-Reasoning in Knowledge Representation," $57,991 5/15/81–1/31/84.

48 297 11 – NSF MCS 79–06110, Paul Purdom/Cynthia Brown, "Average Time Analysis of Algorithms for NP-Complete Problems," $65,164 6/15/81–11/30/83.

48 297 12 – NSF MCS 80–09201, Douglas Hofstadter, "Seek-Whence: A Project in Pattern Understanding," $40,003 7/1/81–12/31/82.

48 297 02 – NSF MCS 79–04183, Mitchell Wand/Daniel Friedman, "Algebraic and Logical Semantics of Computation," $172,437 11/15/81–2/29/84 (a continuation of 48 297 05).

48 297 13 – NSF MCS 82–03978, David Wise/Daniel Friedman, "Applicative Programming for Indeterminate Systems," $129,171 6/15/82–11/30/84.

48 297 14 – NSF MCS 83–04821, Edward Robertson, "Computer Research Equipment," $200,000 6/1/83–3/31/85.

48 297 15 – NSF MCS 83–01742, Paul Purdom , "Analysis of Algorithms for NP-Complete Problem," $48,000 7/1/83–6/30/85.

48 297 16 – NSF MCS 83–04567, Daniel Friedman/Christopher Haynes , "An Operational Model of Languages for Coordinated Computing," $125,880 6/15/83–11/30/85.

48 297 17 – NSF MCS 83–03325, Mitchell Wand/Daniel Friedman/Will Clinger, "Semantic Issues in Computation," $246,350 9/15/83–2/28/86.

48 297 18 – NSF DCR 84–05241, David S. Wise/Steven D. Johnson, "Methods and Architectures for Applicative Programming," $81,937 8/1/84–7/31/85, and $82,818 8/1/85–7/31/86.

53 297 01 – AFOSR 84-0372, John Franco, "Probabilistic Analysis of Algorithms for NP-Complete Problems," $57,285 9/30/84–9/30/85.

54 297 01 – SDF#318 (System Dev. Fdn.), Daniel Friedman, "Coordinated Computing: Tools and Techniques for Distributed Software," $3,000 12/1/82–1/31/83.

56 297 01 – Unico Inc., Edward L. Robertson, "Realtime Semantics," $32,622 9/4/84–5/31/85.

22 402 54 – Dean of Faculties Office: Multidisciplinary Seminar, Stan C. Kwasny/John A. Barnden/C. Ault (Educ.)/N. John Castellan (Psych.)/J. Dunning (Geol.), "Expert Systems and the Analysis of Problem Solving," $12,500 7/1/84–6/30/85.

– Research and Graduate Development, John A. Barnden, "Grant for use of Cyber 205 at Purdue University," $2,000 7/1/84–7/1/85.

– GTE Laboratories, Inc., Paul W. Purdom, Jr., "Industrial Undergraduate Research Participation Program," $18,031 1/1/85–10/1/85.

Corporate Sponsors:

Hewlett-Packard

Procter & Gamble

E.I. duPont de Nemours & Co.

Caterpillar Tractor

June 5, 1985

Pending Grant Proposals

AFOSR, John Franco, "Probabilistic Analysis of Algorithms for NP-Complete Problems," $59,977 (1 year).

NSF Unit: Theoretical Computer Science & Computer Systems Design, George Epstein, "n-Valued Orthogonal Functions and Digital Transforms," $91,299 (2 years).

NSF Unit: Science & Technology to Aid the Handicapped (STAH), Charles Watson, Diane Kewley-Port, Daniel Maki, and Mary Elbert, "Indiana Speech Training Aid," $339,678 (3 years).

NSF Unit: Software Systems Science, Daniel P. Friedman & Christopher T. Haynes, "Liberating and Constraining Control," $222,528 (2 years).

NSF Unit: Computer Research, Steven D. Johnson, William D. Clinger, Franklin Prosser, and David E. Winkel, "Algorithm Driven Hardware Design," $279,891 (2 years).

NSF Unit: Software Systems Science, Mitchell Wand & Daniel P. Friedman, "Semantics of Computation," $311,759 (2 years).

NSF Unit: Computer Systems Design, John O'Donnell, "Applicative Digital Design," $122,878 (2 years).

NSF, Data Support Services Section, Edward L. Robertson, Diane Kewley-Port, Robert Port, Stan Kwasny, and John Barnden. "Computer Science & Computer Engineering Research Equipment," $99,337 (1 year).

NSF Unit: Computer Science Section, Edward L. Robertson, "Computer Science and Computer Engineering Research Equipment," $339,233 (1 year).

NASA: Graduate Student Researcher's Program, Timothy R. Bridges, $15,833 (1 year).

NASA: Computational Investigations Utilizing the MPP, John A. Barnden, "Diagrammatic Information-Processing in Neural Arrays".

NASA: Computational Investigations Utilizing the MPP, John O'Donnell, "Simulating an Applicative Programming Storage Architecyure Using the NASA MPP".

A. Publications Appearing May, 1982–Apr., 1985

Barnden, John A. "The Integrated Implementation of Imaginal and Propositional Data Structures in the Brain" *Proceedings 4th Annual Conference of the Cognative Science Society*, August, 1982.

Barnden, John A. "On Association Techniques in Neural Representation Schemes" *Proceedings 5th Annual Coference of the Cognitive Science Society*, May 1983.

Barnden, John A. "Intensions as Such: An Outline," *Proceedings 8th Int-Joint Conference on Artificial Intelligence*, August, 1983.

Barnden, John A. "On short-term information processing in connectionist theories," *Cognition and Brain Theory, 7* (1), 1984.

Barnden, John A. "Pattern-recognition in a pattern-based neurophysiological model of short-term information-processing," *6th European Conference on Artificial Intelligence,* Pisa, Italy, 1984.

Barnden, John A. "Diagrammatic short-term information processing by neural mechanisms," *Cognition and Brain Theory, 7* (3 and 4), 1984.

Brown, Cynthia A., and Paul W. Purdom, Jr. "A methodology and notation for compiler front end design," *Software - Practice and Experience 14*, (1984), 335–346.

Clinger, William D. "Nondeterministic call by need is neither lazy nor by name," *Proceedings of the 1982 ACM Symposium on Lisp and Functional Programming*, August 1982, 226–234.

Clinger, William D., Daniel P. Friedman, and Mitchell Wand, "A Scheme for a Higher-Level Semantic Algebra" *Proc. US-French Seminar on the Application of Alegrbra to Language Definition and Compilation*, Fontainebleau, France, June, 1982.

Clinger, William D. and C. Halpern. "Alternative semantics for McCarthy's amb," *Proc. of the 1984 NSF/SRC Workshop on Semantics of Concurrency and Nondeterminism.*

Clinger, William D. "The Scheme 311 compiler: an exercise in denotational semantics," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming*, Austin, TX, August 5–8, 1984, 356–364.

Epstein, George with Y–W. Liu, "Positive Multiple-valued Switching Functions — An Extension of Dedekind's Problem," *Proceedings of the Twelfth International Symposium on Multiple-valued Logic,* (May, 1982).

Epstein, George. "The Underlying Ground for Hypothetical Propositions," *Logic in the 20th Century, Scientia,* (1983).

Epstein, George with D.M. Miller and J.C. Munzio, "The Simplification of Multiple-Valued Symmetric Functions" Proceedings 13th International Symposium on Multiple-Valued Logic, May, 1983.

Epstein, George with A. Horn, "Core points in Double Heyting Algebras and Dissectable Latrices" Algebra Universlis, 16, 1983, 204–218.

Epstein, George, with J. Lee and P. O. Mandayam. "Some Observations on n-valued Disjointly Separable Functions," *Proceedings of the Fourteenth International Symposium on Multiple-valued Logic*, (May, 1984).

Epstein, George. "The Enumeration of Monotone Increasing Functions," *1984 Midwest Theory of Computation Symposium*, (Dec., 1984).

Franco, John, "Probabilistic Analysis of the Pure Literal Heuristic for the Satisfiability Problem," *Annals of Operations Research 1* (1984) 273–289.

Friedman, Daniel P. with R. E. Filman, "Models, Languages and Heuristics for Distributed Computing" National Computer Conference, June 1982 AFIPS Conf. Proceeding, AFIPS Press, Arlington, VA (1982) 671–677.

Friedman, Daniel P., Christopher T. Haynes, and Eugene Kohlbecker, "Programming with Continuations," *Program Transformation and Programming Environments,* (P. Pepper, ed.), Springer-Verlag Berlin Heidelberg (1984) 263–274.

Friedman, Daniel P. and Mitchell Wand. "Reification: Reflection Without Metaphysics," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 348–355.

Friedman, Daniel P. and Christopher T. Haynes, "Constraining Control," *Conf. Record of the 12th Annual ACM Sym. on Principles of Programming Languages*, New Orleans, LA. (Jan. 1985), 245–254.

Haynes, Christopher T. and Daniel P. Friedman, "Engines Build Process Abstractions," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 18–24.

Haynes, Christopher T., Daniel P. Friedman, and Mitchell Wand. "Continuations and Coroutines," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 293–298.

Haynes, Christopher T. "A Theory of Data Type Representation Independence," *Proceedings of Semantics of Data Types*, (eds.) G. Kahn, D. MacQueen, and G. Plotkin, L. N. C. S. 173, Springer-Verlag, Berlin, (1984), 157–175.

Johnson, Steven D. "Circuits and Systems Implementing Communication with Streams," *Proceedings of the 10th IMACS World Congress on Systems Simulation and Scientific Computation* (Aug. 1982) W.F. Ames & R. Vichnevetsky (eds.) Vol. 5 *Parallel & Large Scale Computers: Systems, Applications & Performance Evaluation.*

Johnson, Steven D. "Synthesis of Digital Designs from Recursion Equations," The ACM Distinguished Dissertation Series, MIT Press, 1984.

Johnson, Steven D. "Applicative Programming and Digital Design." *Conference Record of 11th ACM Symposium on Principles of Programming Languages* (Jan. 1984), 218–227.

Epstein, George with A. Horn, "Core points in Double Heyting Algebras and Dissectable Latrices" Algebra Universlis, 16, 1983, 204–218.

Epstein, George, with J. Lee and P. O. Mandayam. "Some Observations on n-valued Disjointly Separable Functions," *Proceedings of the Fourteenth International Symposium on Multiple-valued Logic*, (May, 1984).

Epstein, George. "The Enumeration of Monotone Increasing Functions," *1984 Midwest Theory of Computation Symposium*, (Dec., 1984).

Franco, John, "Probabilistic Analysis of the Pure Literal Heuristic for the Satisfiability Problem," *Annals of Operations Research 1* (1984) 273–289.

Friedman, Daniel P. with R. E. Filman, "Models, Languages and Heuristics for Distributed Computing" National Computer Conference, June 1982 AFIPS Conf. Proceeding, AFIPS Press, Arlington, VA (1982) 671–677.

Friedman, Daniel P., Christopher T. Haynes, and Eugene Kohlbecker, "Programming with Continuations," *Program Transformation and Programming Environments,* (P. Pepper, ed.), Springer-Verlag Berlin Heidelberg (1984) 263–274.

Friedman, Daniel P. and Mitchell Wand. "Reification: Reflection Without Metaphysics," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 348–355.

Friedman, Daniel P. and Christopher T. Haynes, "Constraining Control," *Conf. Record of the 12th Annual ACM Sym. on Principles of Programming Languages*, New Orleans, LA. (Jan. 1985), 245–254.

Haynes, Christopher T. and Daniel P. Friedman, "Engines Build Process Abstractions," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 18–24.

Haynes, Christopher T., Daniel P. Friedman, and Mitchell Wand. "Continuations and Coroutines," *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming,* Austin, TX. (Aug., 1984) 293–298.

Haynes, Christopher T. "A Theory of Data Type Representation Independence," *Proceedings of Semantics of Data Types*, (eds.) G. Kahn, D. MacQueen, and G. Plotkin, L. N. C. S. **173**, Springer-Verlag, Berlin, (1984), 157–175.

Johnson, Steven D. "Circuits and Systems Implementing Communication with Streams," *Proceedings of the 10th IMACS World Congress on Systems Simulation and Scientific Computation* (Aug. 1982) W.F. Ames & R. Vichnevetsky (eds.) Vol. 5 *Parallel & Large Scale Computers: Systems, Applications & Performance Evaluation*.

Johnson, Steven D. "Synthesis of Digital Designs from Recursion Equations," The ACM Distinguished Dissertation Series, MIT Press, 1984.

Johnson, Steven D. "Applicative Programming and Digital Design." *Conference Record of 11th ACM Symposium on Principles of Programming Languages* (Jan. 1984), 218–227.

Kwasny, Stan C., "Ill-Formed and Non-Standard Language Problems," position paper, Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics, June, 1982, 164–166.

Kwasny, Stan C., with Jonathan Dalby and Robert Port. "Rules for Automatic Mapping between Fast and Slow Speech," *Kentucky Foreign Language Conference,* April, 1984.

O'Donnell, John, "Charles Babbage: Pioneer of the Computer," extended review, *American Mathematical Monthly*, November, 1984.

Purdom, Paul W. and Cynthia A. Brown, "Searching in Polynomial Average Time," Twentieth Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill. (1982).

Purdom, Paul W., Jr. and Cynthia A. Brown, "Evaluating Search Methods Analytically," National Conference on Artificial Intelligence (1982) 124–127.

Purdom, Paul W. with Allen Goldberg and Cynthia A. Brown, "Average Time Analysis of Simplified Davis–Putnam Procedures," *Information Processing Letters 15* (1982), 72–75.

Purdom, Paul W. and Cynthia A. Brown. "An Analysis of Backtracking with Search Rearrangement," *SIAM J. on Computing 12* (1983), 717–733.

Purdom, Paul W. "Search Rearrangement Backtracking and Polynomial Average Time," *Artificial Intelligence 21* (1983), 117–133.

Purdom, Paul W. "Solving Satisfiability Problems with Less Searching," *IEEETPAMI 6* (1984), pp 510–513.

Purdom, Paul W. and Cynthia A. Brown. "A Methodology and Notation for Compiler front End Design," *Software - Practice and Experience 14*, (1984), 335–346.

Robertson, Edward L. with K. Keutzer, "The M-Shuffle as an interconnection network for SIMD machines," *Proc. 20th Ann. Allerton Conference on Communication, Control, and Computing.* (1982).

Robertson, Edward L. and Johann P. Malmquist. "On the Complexity of Partitioning Sparse Matrix Representations," *BIT 24* (1984), 60–68.

Wand, Mitchell, "Deriving Target Code as a Representation of Continuation Semantics" ACM Trans. on Prog. Lang. & Systems 4 (1982) 496–517.

Wand, Mitchell. "Loops in Combinator-Based Compilers," *Conf. Rec. 10th ACM Symposium on Principles of Prog. Lang.*, Austin, Texas, January, 1983. Also in *Information and Control 57*, 2–3, (May/June, 1983), 148–164.

Wand, Mitchell. "What is Lisp?" *American Mathematical Monthly 91*, (1984), 32–42

Wand, Mitchell. "A Types-as-Sets Semantics for Milner-style Polymorphism" *Conf. Rec. 11th ACM Symp. on Principles of Prog. Lang.* (1984), 158–164.

Wand, Mitchell. "Semantic Prototyping System" *Proc. ACM SIGPLAN '84 Symposium on Compiler Construction* (1984), 213–221.

Wand, Mitchell. "Embedding Type Structure in Semantics" *Conf. Rec. 12th ACM Symp. on Principles of Prog. Lang.* (1985), 1–6.

Wise, David S. "Interpreters for Functional Programming," In Darlington, J., Henderson P., and Turner, D. (eds.) *Functional Programming and its Applications*, Cambridge University Press (1982).

Wise, David S. "Functional Programming,". In A. Ralston (ed.), *Encyclopedia of Computer Science*, New York, Van Nostrand Reinhold (1983), 647–650.

Wise, David S. "Representing Matrices as Quadtrees for Parallel Processors: Extended Abstract." *SIGSAM Bulletin 18* (3):24–25 (August, 1984).

Wise, David S. "The Applicative Style of Programming," *ABACUS, 2* (2), (1985), 20–32.

B. Articles to Appear

Clinger, Willaim D., Daniel P. Friedman, and Mitchell Wand. "A Scheme for a Higher-level Semantic Algebra," May, 1983, to appear in *Algebraic Methods in Semantics*, (J. Reynolds & M. Nivat, eds.), Cambridge University Press.

Chouknane, E. and John Franco, "An Approximation Algorithm for the Maximum Independent Set Problem on Cubic Planar Graphs," to appear in *Networks.*

Epstein, George and R. R. Loka. "Block Functions-A New Class of Sequency Preserving n-valued Orthogonal Functions," *Proceedings of the Fifteenth International Symposium on Multiple-valued Logic.*

Epstein, George and R. R. Loka. "Extensions of Block Functions," *Proceedings of the Fifteenth International Symposium on Multiple-valued Logic.*

Hall, Cordelia V. and John O'Donnell. "Debugging in s Side Effect Free Programming Environment." To appear in *Proceedings of the 1985 ACM SIGPLAN Symposium on Programming Environments.*

O'Donnell, John. "A Basis for Constructing Programming Environments," To appear in *Proceedings of the 1985 ACM SIGPLAN Symposium on Programming Environments.*

Purdom, Paul W. and Cynthia Brown. "The Pure Literal Rule and Polynomial Average Time," (to appear in *SIAM J. on Computing*).

Purdom, Paul W. and Cynthia Brown, "Fast Many-to-One Matching Algorithm." to appear in *First International Conference on Rewriting Techniques and Applications"*.

Robertson, Edward L., I. Munro, and R. Karlson, "The nearest neighbor problem on bounded domains," *Automata, Languages, and Programming. Proceedings 1985*, Lecture Notes in Computer Science, Springer-Verlag (1985), (to appear).

Wand, Mitchell, with A. Meyer. "Continuation Semantics in Typed Lambda-Calculi," *Logics of Programs* (Brooklyn, June, 1985), (R. Parikh, ed.) *Springer Lecture Notes in Computer Science, vol. 193* (1985), 219–224.

Wand, Mitchell. "Lambda Calculus," *Encyclopedia of Artificial Intelligence*, (S. C. Shapiro, ed.) Wiley-Interscience, (to appear).

Wise, David S. "Representing Matrices as Quadtrees for Parallel Processors," *Information Processing Lett.* (to appear).

Wise, David S. "Design for a Multiprocessing Heap with On-Board Reference Counting," *Conf. Proceedings Functional Programming Languages and Computer Architecture,* Nancy, France (1985)

C. Books in Print

Epstein, G. *Modern Uses of Multiple-valued Logic*, Reidel, 1977 (Co-editor with J.M. Dunn).

Friedman, D.P. *The Little LISPer*, Science Research Associates, Palo Alto, 1974.

Friedman, D.P. and R. E. Filman,*Coordinated Computing: Tools and Techniques for Distributed Software*. McGraw-Hill (February, 1984), 370.

Hofstadter, D.R. *Gödel, Escher, Bach: an Eternal Golden Braid*, Basic Books, New York, 1979.

Hofstadter, D.R. and Dennett, D. (eds.) *The Mind's I*, Basic Books, New York, 1981.

Purdom, Paul W. and Cynthia Brown. *Analysis of Algorithms*, Holt, Rinehart & Winston, N.Y., 1985.

Wand, M. *Induction, Recursion, and Programming*, Elsevier North Holland, New York, 1980.

Winkel, D. and Prosser, F. *The Art of Digital Design: An Introduction to Top-Down Design*, Prentice-Hall, Englewood Cliffs, N. J., 1980.

D. Other Technical Reports

TR 126 – Frank Prosser and Patricia J. Brajnikoff, Guide to the Printed Circuit Board Fabrication Facility. (1982).

TR 127 – George Epstein, Switching Theory, Multiple-Valued Logic and Logic Design (January, 1983)

TR 130 – Douglas Hofstadter, Who Shoves Around Inside the Careenium? or What is the meaning of the word (July, 1982).

TR 131 – Barnden John A., Continuum of Diagrammatic Data Structures in Human Cognition. (October, 1982)

TR 132 – Douglas R. Hofstadter, (Artificial Intelligence: Subcognition as Computation. (November, 1982)

TR 134 – Mitchell Wand, Research in the Computer Science Department at Indiana University, 1981-1982. (December, 1982)

TR 135 – Chun-Hung Tzeng and Paul W. Purdom, Jr., A Theory of the Heuristic Game Tree Search. (December, 1982)

TR 136 – Douglas Hofstadter, Metafont, Metamathematics and Metaphysics. (December, 1982)

TR 137 – W. D. Clinger, C. Fessender, D. P. Friedman, and C. T. Haynes, Scheme 311 Reference Manual. (February, 1983).

TR 138 – Pee-Hong Chen, Implementation of Two Data-Flow Models in Scheme. (February, 1983).

TR 139 – Dave Laymon, Scheme Translation of Functions from *Functional Programming Application and Implementation.* (Spring, 1983).

TR 140 – Jim Burns, K. Keutzer, and Paul W. Purdom, Communication by Non-Writing in Synchromes Parallel Machines. (June, 1983)

TR 142 – David S. Wise "A Powerdomain Semantics for Indeterminism," (July, 1983), Revised: (Jan. 1984).

TR 143 – Khaled Bugara and Cynthia Brown. On the Average Case Analysis of Some Satisfiability Model Problems.

TR 144 – Edward L. Robertson and J. P. Malmquist. On the Complexity of Partitioning Sparse Matrix Representations.

TR 145 – Chen, Chi, Ost, Sabbagh, Springer. Scheme Graphics Reference Manual. (August, 1983).

TR 146 – Pee-Hong Chen and David Sabbagh. A Functional Approach to Geometric Applications in Computer Graphics. (August 1983).

TR 147 – Daniel P. Friedman, with P. Chen. Prototyping Data Flow by Translation into Scheme. (Aug. 1983)

TR 148 – Mitchell Wand. A Semantic Algebra for Logic Programming (August, 1983).

TR 149 – Kent Dybvig. C-Scheme Reference Manual. (September, 1983).

TR 150 – Marek J. Lao. A Case Study of a Combinator-Based Compiler for a Language with Blocks and Recursive Functions. (October, 1983).

TR 153 – Daniel P. Friedman, C. T. Haynes, E. E. Kohlbecker and M. Wand. Scheme 84 Reference Manual. (Feb. 1984).

TR 154 – George Epstein, J. Lee, and P.O. Mandayam. Some Observations on n-Valued Disjointly Separable Functions. (March 9, 1984).

TR 155 – Eugene Kohlbecker. eu-Prolog: Reference Manual and Report. (April, 1984).

TR 156 – Marek J. Lao. Direct Denotational Semantics: Combinator-based Compilation of Goto's. (May, 1984).

TR 157 – Eugene Kohlbecker. Using mkmac. (May 1984).

TR 160 – C.D. Halpern. An Implementation of 2-Lisp. (June, 1984).

TR 162 – Mitchell Wand and S. Kolbl. Linear Future Semantics and its Implementation (Oct., 1984).

TR 164 – John Franco and Ming-Te Chao. Probabilistic Analysis of the Unit Clause and Maximum Occurring Literal Selection Heuristics for the 3-Satisfiability Problem. (Dec. 1984).

TR 165 – John Franco and Ming-Te Chao. Probabilistic Analysis of a Generalization of the Unit Clause Literal Selection Heuristic for the k-Satisfiability Problem. (Jan. 1985).

TR 166 – L. David Sabbagh and Pee-Hong Chen. Scheme as an Interactive Graphics Programming Environment. (Feb. 1985).

TR 167 – John Franco. On the Probabilistic Performance of Algorithms for the Satisfiability Problem. (Mar. 1985).

TR 168 – Steven D. Johnson. Storage Allocation for List Multiprocessing. (Mar. 1985).

Colloquia Visitors
1984-85

Simon L. Peyton Jones
Department of Computer Science, University College London
Arbitrary Precision Arithmetic Using Continued Fractions
August 17, 1984

John Hughes
Programming Research Group, Oxford
Functional Languages and Parallel Computers
August 21, 1984

Mary Sheeran
Programming Research Group, Oxford
muFP - A VLSI Design Language and its Use in the Design of Systolic Arrays
August 20, 1984

Adam Bojanczyk
Centre for Mathematical Analysis, Austrailian National University
Systolic Algorithms in Numerical Linear Algebra
August 30, 1984

Margaret Montenyohl
Indiana University
Report on International Summer School on Control Flow and Data Flow: Concepts in Distributed Programming
September 11, 1984

David Oran
Digital Equipment Corporation, Distributed Systems Architecture
Three Network Architectures and Their Transport Protocols
September 18, 1984

Doug Hofstadter
University of Michigan/Indiana University
Analogies and Roles in Human and Machine Thinking or Why I am Going to be in a Psychology Department
October 18, 1984

David C. Brown
Department of Computer & Information Science, Ohio State University
Expert Systems for a Class of Mechanical Design Activity
October 25, 1984

Eugene Kohlbecker
Indiana University
Seminar and Report on a Scheme Workshop
October 30, 1984

Megha Shyam
Hewlett-Packard
VLSI Design
November 6, 1984

Catherine Smith
Datacom Laboratory, Hewlett-Packard
Software Engineering and Hewlett-Packard
November 7, 1984

Graeme Hirst
Computer Science Department, University of Toronto
Natural Language Semantic Interpretation Against Ambiguity
November 15, 1984

George Epstein
Indiana University
Some Open Problems in the Enumeration of Monotone Increasing Functions
November 27, 1984

Bob Paige
Computer Science Department, Rutgers University
Stream Processing in Code Optimization
December 4, 1984

Christopher Haynes
Indiana University
Constraining Control
December 11, 1984

Reid G. Simmons
Artificial Intelligence Laboratory, MIT
Problem Solving in Geologic Interpretation
January 8, 1984

David S. Wise
Indiana University
Applicative Programming and Architectures for Multiprocessing
January 29, 1985

David S. Wise
Indiana University
Architecture Derived from an Applicative Programming Language
January 31, 1985

Paul Purdom
Indiana University
An Introduction to Rewrite Rules
February 5, 1985

Gary L. Peterson
University of Rochester
Efficient Algorithms for Elections in Meshes and Complete Networks
January 22, 1985

Marvin Minsky
MIT
Patten Foundation Visit
February 18 through February 20, 1985

Dennis Gannon
Purdue University
On The Structures of Multilevel Parallel in Scientific Computation
February 22, 1985

Kent Dybvig
University of North Carolina at Chapel Hill
A Retargetable Native-code Compiler for Scheme
March 1, 1985

Simon Kasif
University of Maryland
And/Or Parallelism in Logic Programs
March 4, 1985

Robert P. Futrelle
University of Illinois at Urbana-Champaign
Automating Science: Getting a Computer to Read the Literature
March 21, 1985

Laxmikant V. Kale
State University of New York
Parallel Architectures for Problem Solving in the Framework of Logic Programming
March 20, 1985

Ganesh Gopalakrishnan
SUNY at Stony Brook
Formal Specification and Automation of VSLI Designs
March 25, 1985

Paul W. Abrahams
Challenges in Programming Language Design
March 26, 1985

Stanley Jefferson
University of Illinois
Execution of Final Algebra Specifications
March 28, 1985

David MacQueen
Bell Laboratories, Murray Hill, NJ
Modules in ML
April 2, 1985

John Van Rosendale
ICASE NASA Langley, Hampton, VA
Blaze: A Parallel Language for Scientific Programming
April 12, 1985

Ahmed Elmagarmid
Department of Computer and Information Science
The Ohio State University
Deadlock Detection in Distributed Processing Systems
April 15, 1985

Dirk Van Gucht
Department of Computer Science
Vanderbilt University
Non-First-Normal-Form Relational Databases
April 25, 1985

Kent Dybvig
Data General Corporation
Research Triangle Park, NC
Scheme on a Cellular Computer
April 29, 1985

Steve Bellenot
Departments of Mathematics and Computer Science
Florida State University
"Time Warp on a Hypercube"
May 10, 1985

Jack Lutz
Department of Mathematics
California Institute of Technology
Logical and Computational Complexity on Finite Structures
May 13, 1985

Paul W. Purdom
Department of Computer Science
Indiana University
"Fast Many-to-One Matching Algorithms"
May 16, 1985

James E. Burns
Department of Computer Science
Indiana University
The Byzantine Firing Squad Problem
May 23, 1985

John T. O'Donnell
Department of Computer Science
Indiana University
Dialogues: A Basis for Constructing Programming Environments
May 30, 1985

Cordelia V. Hall
John T. O'Donnell
Computer Science Department
Indiana University
Debugging in a Side Effect Free Programming Environment
May 30, 1985

Trevor Vickers
Department of Computer Science
University of New South Wales
Kensington NSW 2033 AUSTRALIA
Quokka: A Translator Generator Using Denoatational Semantics
July 10, 1985

Colloquia Visitors
1983-84

Douglas Hofstadter
Indiana University
HAN ZI: A Program for Generating Chinese Characters in Many Different Styles
September 13, 1983

Joe Anderson
Naval Weapons Support Center, Crane
Computer Aided Design Verification
September 29, 1983

David Sabbagh
Indiana University
Some Numerical Techniques for Inverse Problems
October 22, 1983

J.A. Kalman
(Visiting the Mathematics Department)
Indiana University, Bloomington
Applications of Mechanical theorem Proving to Nonclassical Logic
November 10, 1983

Christopher Haynes
Indiana University
LISP: What's Right, What's Wrong, and What to do About it
November 15, 1983

Jonathan Bein
Martin Marietta Denver Aerospace
Control in a goal Directed Production System
November 22, 1983

John Althauser
Indiana University
"Computers and Electronic News Delivery"
November 19, 1983

Steven Johnson
Indiana University
Applicative Programming and Digital Design
January 5, 1984

Mitchell Wand
Indiana University
A Types-as-Sets Semantics for Milner-Style Polymorphism
January 9, 1984

Vladimir Vrecion
Faculty of Law, Charles University, Prague, Czechoslovakia
Artificial Intelligence Systems for Normative Information
January 12, 1984

George Epstein
Indiana University
Decision Logics
January 31, 1984

A Videotape Movie
Produced by Knowledge System Area, XEROX PARC
Knowledge Programming in Loops
February 10, 1984

Harry Quachenboss, Calvin Troupe, Jim Stevenson, Charlie Ford
Honeywell, Inc.
The Multics Operating System
February 14, 1984

Romane Clark
Department of Philosophy, Indiana University
A Philosopher's View of Puzzle-Solving
February 14, 1984

Richard B. Kieburtz
Oregon Graduate Center
Marigold --A Functional, Flow-graph Language
February 23, 1984

Stephen Nemecek
University of Southwestern Louisiana
A Standard Form of Dataflow Graphs
February 24, 1984

Chris Haynes
Indiana University
Scheme 84: A Language for Meta-Programming *Including* How to Meta-Program an Operating System with Engines
March 6, 1984

David Schmidt
Kansas State University
Detecting Global Variables in Denotational Specifications
March 9, 1984

Lawrence Wos
Mathematics and Computer Science Division
Argonne National Laboratory
Automated Reasoning and its Applications
March 20, 1984

John Franco
Case Western Reserve University
Probabilistic Analysis of Algorithms for the Satisfiability Problem
March 23, 1984

Dana Richards
Computer Science, IUPUI
Finding Short Cycles
March 28, 1985

Steven Wartik
TRW Defense Systems Group
A Multi-Level Approach to the Production of Requirements for
Interactive Computer Systems
April 10, 1984

Rishiyur Nikhil
University of Pennsylvania
Functional Programming Languages and Databases
April 17, 1984

Mark Fulk
SUNY at Buffalo
Machine Inductive Inference of Recursive Functions
April 20, 1984

Burkhard Monien
Visitor to Dept. of Elec. Engineering & Computer Science,
Northwestern University
Improved Worst-Case Bounds for NP-Complete Problems
April 18, 1984

Virginia Lo
Computer Science Department, University of Illinois
Task Assignment in Distributed Systems
April 24, 1984

Richard Salter
Oberlin College
Getting A.I. Planning Systems to Reason about Time
May 10, 1984

Terry Weymouth
Computer and Information Science Dept., University of Massachusetts
Using Object Descriptions in a Schema Network for Machine Vision
May 21, 1984

John T. O'Donnell
Computer Science Department
Indiana University
Debugging in a Side Effect Free Programming Environment
May 30, 1985

Trevor Vickers
Department of Computer Science
University of New South Wales
Kensington NSW 2033 AUSTRALIA
Quokka: A Translator Generator Using Denoatational Semantics
July 10, 1985

Colloquia Visitors
1982-83

Neil Jones
University of Copenhagen
Control Flow Treatment in a Simple Semantics-Directed Compiler Generator
August 12, 1982

Paul Purdom
Indiana University
TeX at IU
September 7, 1982

Peter Wallis
University of Bath
ADA and Portable Programming
September 16, 1982

John O'Donnell
Indiana University
Computer Architecture Support for List Processing Languages
September 28, 1982

Dana Nau
University of Maryland
The Nature of Pathology in Game Trees
October 22, 1982

Mark Johnson
Southern Illinois University
Metaphorical Understanding as a Problem for Cognitive Science
November 12, 1982

Peter Suber
Earlham College
Reflexivity and the Law
December 2, 1982

Clem McDonald, M.D.
Regenstrief Health Center, Indpls.
Computer Reminders in Medicine
December 7, 1982

Mitchell Wand
Indiana University
Loops in Combinator-Based Compilers
January 19, 1983

Chun-Hung Tzeng
Indiana University
"A Theory of Heuristic Game Tree Search"
February 1, 1983

Frank Oles
Syracuse University
The Nature of Denotation for Algol-like Languages
February 11, 1983

Franklin Prosser
Indiana University/Univ. of Wyoming
The Logic Engine
February 8, 1983

Robert Webber
University of Maryland
Using Quadtrees to Represent Geographical Data
March 1, 1983

Deborah Joseph
University of California, Berkeley
Arms, Centipedes and Pianos: How hard are they to move in close Quarters?
March 7, 1983

Sam Kim
University of Minnesota
Characterizations and Computational Complexity of Some Models for Parallel Computation
March 28, 1983

Seymour Goodman
University of Arizona
A Perspective on Computing in the Soviet Union
March 29, 1983

Michael Snider
Battelle, Columbus Laboratories
Software Engineering Today
April 12, 1983

Walter Schnyder
ETH-Zurich
Canonical Forms for Graphs of Bounded Valence
April 15, 1983

Cynthia Brown
GTE Laboratories
Symbol Table Operations for Explicit Scope Control in a Separate Compilation Environment
April 22, 1983

Daryel Sachse-Akerlind
Australian National University
Anomalous Algorithm and Provable Complexity Bounds
May 26, 1983

Bob Baron
University of Iowa
Neural Mechanisms for Visual Imagery
June 2, 1983

Steve Johnson
Indiana University
DSI/DAISY Boxes, Arrows, and Clouds in 1983 (!?)
July 7, 1983

David Wise
Indiana University
A Powerdomain Semantics for Indeterminism
July 8, 1983

Indiana University
Department of Computer Science
1984

Faculty:	20 FTE (about 21 individuals)
Staff:	1 Administrative-Professional
	6 Clerical
	1 Programmer
	1 Engineer
	1 Advisor (Undergraduate)
Students:	~ 400 Undergraduate Majors (GPA for Admission to major is 3.2) 154 Graduate Majors
Degrees Granted:	40 B.S.
(1984)	61 B.A.
	70 M.S.
	4 Ph.D. (Since program began in 1981)
Student Support:	$6210 1984-85 stipend
	49 Associate Instructors (teaching assistants)
	8 Research Assistants

Budget:

Academic Salaries	710,210	
Associate Instructors	163,990	plus supplements
Staff Salaries	156,695	
Supplies & Expenses	73,660	plus supplements

Indiana University
Department of Computer Science
1983

Faculty:	18 FTE (about 18 individuals)
Staff:	1 Administrative-Professional/Advisor
	4 Clerical
	1 Programmer
	1 Engineer
Students:	~ 642 Undergraduate Majors (GPA for Admission to major is 3.4)
	152 Graduate Majors
Degrees Granted: (1983)	33 B.S.
	71 B.A.
	109 M.S.
	2 Ph.D. (Since program began in 1981)
Student Support:	$6000 1982-83 stipend
	$6000 1983-84 stipend
	51 Associate Instructors (teaching assistants)
	8 Research Assistants

Budget:

Academic Salaries	640,534	
Associate Instructors	128,444	plus supplements
Staff Salaries	115,749	
Supplies & Expenses	71,411	plus supplements