Research in the Computer Science Department

at Indiana University

1981-82

edited by

Mitchell Wand

Computer Science Department

Indiana University

Bloomington, Indiana 47405

TECHNICAL REPORT NO. 134

RESEARCH IN THE COMPUTER SCIENCE DEPARTMENT at INDIANA UNIVERSITY

1981-82

Edited by

Mitchell Wand

December, 1982

Table of Contents

§1 Faculty

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

Cynthia Brown, Assistant Professor; Ph.D., University of Michigan. Analysis of algorithms, compilers.

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., MIT. Semantics of programming languages, artificial intelligence, nondeterministic concurrent computation, logic.

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

Robert Filman, Assistant Professor; Ph.D., Stanford University. Artificial Intelligence, programming languages, distributed computing, data base systems, logic, and man-machine interaction.

Daniel Friedman, Associate Professor; Ph.D., University of Texas at Austin. Program methodology, formal semantics, artificial intelligence, distributed computing, simulation languages, applicative languages, LISP.

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.

Smith Higgins, Associate Professor; Ph.D., Notre Dame. Computer science education, number theory, modern algebra.

Douglas Hofstadter, Associate Professor; Ph.D., University of Oregon. Artificial intelligence, perception of patterns and style, self-monitoring and self-perception, philosophy of mind.

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

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, Associate Professor; Ph.D., California Institute of Technology. Analysis of algorithms, compilers.

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

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

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

§2 Facilities

The following is a list of equipment which is available purely for research use in the Indiana University Computer Science Department:

VAX 11/780, 500 Mbyte disk, 3.5 Mbyte RAM, running UNIX 4.1bsd
Canon laser beam printer from Imagen Corporation

In addition, the department has other facilities which are shared between research and instructional uses.

Ungermann/Bass Ethernet terminal servers
HP3000 model 44
Motorola 68000 systems - several systems are on hand.

The department also maintains and operates a digital electronics laboratory which includes facilities for constructing and testing hardware designs. The lab supports a range of prototyping activity from wire wrap construction through fabrication of double sided plated through hole printed circuit boards.

In addition, the following University facilities are available to researchers at Indiana University:

DEC KL10
Prime 750
4 VAX 11/780(VMS)
IBM 4341
CDC Cyber 172, soon to be replaced by CDC 170/855
Calcomp, Versatec and Zeta plotters

§3 Summaries of Research Projects

John A. Barnden

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.

Concurrent Programming Using Phases

A concurrent programming language, PhasL, is being developed in which the notions of condition and process are united in a single 'phase' concept, and in which there are no sequential control constructs. A PhasL program defines a self-modifying tree of phases. The language allows elegant solution of complex concurrent-programming problems. There are some interesting implementational and formal-semantical problems for which solutions are being devised. Automatically checkable restrictions of the language to allow distributed implementation are being studied. The modification of PhasL into a form suitable for artificial-intelligence applications is receiving attention.

George Epstein

Multiple-Valued Logic

I continued research in multiple-valued logic design. My two basic areas were in theory and applications. In theory, I studied properties of core points in lattices, with A. Horn. In applications, I studied symmetric functions with J.C. Muzio and D.M. Miller, and positive functions with Y.W. Liu.

Robert E. Filman

Meta-Reasoning in Knowledge Representation

A central issue in the creation of an intelligent computer system over a domain is the representation of the knowledge embedded in that domain in machine-usable forms. This research suggests that it is possible to construct intelligent systems where the underlying facts, the legal conclusions derivable from those facts, and the appropriate control strategy to derive those conclusions can all be expressed in machine-usable form as instances of meta-theoretic reasoning. We are exploring the expression and resolution of knowledge representation problems by a combination of procedural attachment, rewriting systems, and meta-theoretic representations, in the context of a first order logic system. Issues to be addressed include the representation of heuristics and control, hypothetical reasoning, non-monotonic systems, knowledge and belief, time, strategies, learning and game theory.

Robert E. Filman and Daniel P. Friedman

Models and Techniques for Distributed Systems

Currently we are writing a book based on the development of tools and techniques for distributed software. In the text we describe and discuss some 15 languages and models for interprocess communication. In addition, we discuss heuristics for problem solving in the distributed processing environment.

Daniel P. Friedman and William Clinger

Continuation Based Parallel Processing

In giving the standard semantics of a programming language, four concepts are central: syntactic objects (code), environments, continuations, and the store. Closely connected to these semantic concepts are certain operational concepts corresponding to typical implementation techniques. Among them is the concept of a closure, which may be thought of as a code/environment pair. Another is the concept of a process, which may be thought of as an active closure/continuation pair. Processes may be thought of as communicating via locations in the store. Our view of computation emphasizes these connections between semantic and operational concepts.

The programming language Scheme treats closures and continuations as first class objects. Closures have been fairly well exploited, but the art of programming with explicit continuations remains relatively undeveloped. Much as closures seem most useful when combined with side effects (a synergism between the functional and imperative worlds), continuations turn out to be most useful when combined with concurrency (a synergism between the sequential and parallel worlds). Side effects are phenomena that greatly enrich (i.e. complicate) the semantics of closures. Assignment statements are a means for causing such side effects. Analogously, concurrency is a phenomenon that greatly enriches the semantics of continuations. We have invented a new Scheme primitive, par2, for causing concurrency.

We have a straightforward operational semantics for par2, describing its actions in terms of continuations and processes (threads of computation).

We have used this one primitive function, par2, to implement busy waiting implementations of amb, parallel or, and their equivalents; non busy waiting amb, parallel or, and their equivalents; cobegin/coend, fork, kill, kill transitively, vanish, and vanish transitively; by-need amb, active cells, and exchange functions; and with the addition of a fair mutual exclusion primitive, actors.

We intend to investigate further the power of these tools. Specifically, we are exploring the interactions between closures, continuations, and concurrency. We also wish to develop a denotational semantics corresponding to our operational semantics.

Daniel P. Friedman and Christopher Haynes

An Operational Model of Languages for Coordinated Computing

With the proliferation of inexpensive computing devices, there is a clear need for programming languages that provide disciplined and efficient facilities to coordinate their computing. A number of proposals for such languages have been made with widely varying approaches to the problems of distributed computing, yet little is known of their interrelationships.

This research will attempt to develop a precise and coherent description of languages for coordinated computing by providing an operational definition of their semantics in the programming language Scheme. Besides resolving ambiguities in the existing natural language descriptions of these languages, such uniform descriptions clarify the relationships between the proposed languages. Such a clarification would aid in designing future languages for coordinated computing, as well as in making coherent choices between existing languages.

Of immediate interest is the problem of process scheduling, which has traditionally been perceived as a function of the underlying operating system. We have implemented an operational description of the hybrid language Cell. This language embodies a complex scheduling mechanism and a rather unorthodox control structure. This preliminary research indicates that such complexities may be concisely specified in the Scheme language, and suggests that scheduling may be brought into the domain of programming languages in a unified and transparent manner.

Daniel P. Friedman and David S. Wise

Applicative Programming for Systems

This project extends an applicative approach to programming parallel computers for system-level problems which require synchronization. This applicative style requires that all programs be stated as expressions in a calculus, where arguments are input and values are output; sequence of execution or evaluation is irrelevant. Synchronization of a process with an external event—perhaps the output of another processor—is accomplished through the multiset data structure, which encapsulates all the interprocessor contention within a shared data object. We have developed the semantics of multisets and lists at the formal level, but several issues stand between this approach to difficult system-level problems (e.g. airline reservation system) and its direct implementation on arbitrary numbers of communicating processors.

The most important one is fairness. An originally unordered multiset takes on an order as it is probed, but it may also be augmented; fairness requires that convergent values, whether original or augmenting, may not be deferred arbitrarily in that order. Another problem is controlling the propagation of the strict functions which we have introduced in order to restrict convergence of structured values within a multiset. the selection of a storage-management scheme for a heap shared by many processors remains open; generalizations to reference counting or restrictions to garbage collection are needed to constrain memory conflicts. Several other problems, including the role of functions as values, and the strategies for firmware implementation of multisets, will be considered.

Douglas R. Hofstadter

Creative Perception

With my graduate students I have been working on one central, general, abstract problem, and on three particular examples of it, each one in a different domain. The general problem might be called "creative perception", or "perception with abstraction built into it". These phrases emphasize the well-known fact that perception is not direct, but requires considerable interpretation, or addition of information. We feel that perception in a general sense is the central and only problem of artificial intelligence. This requires construing perception in the broad way we are stressing.

Our three projects are:

(1) Seek-Whence, operating in the domain of sequences — infinite sequences of integers grouped into patterns. In this project, our aim is to imitate the cognitive processes carried out by a human who watches a pattern appear slowly (as successive terms of the sequence are revealed): forming guesses, seeing them supported or undermined, reforming those guesses or making new ones, and eventually coming to a firm conclusion — thus, "seeking whence" the sequence comes. In effect, we are studying the scientific process as it occurs in a miniworld. We are not nearly as concerned with simply reproducing the ability to get the right answer after seeing many terms as with making a program capable of being in all the typically human intermediate states of guessing, confusion, hope, disappointment, and so forth, as arise after seeing only a handful of terms. We are also concerned with one other aspect of doing science that we see as absolutely crucial: the ability to manufacture what we call "variations on a theme".

(2) Letter Spirit, whose concern is the understanding and modeling of the method by which people can see sample letterforms and "get the hang of the style", so that they can go on and complete an entire alphabet in "the same style". The crucial thing is to abstract out some "essence of style" that can be transported and implanted in any letter.

(3) Jumbo, a program that takes a group of letters and shuffles them about in an attempt to form "English-like" structures, in imitation of someone playing the newspaper game called "Jumble". The central idea is to allow exploration of the space of possibilities without having to manufacture each candidate from scratch, but rather one simply knows how parts can regroup and reorganize to form coherent wholes. Thus, one perceptual whole can jump directly into another perceptual whole without being totally taken apart.

In all three of these projects, we have been guided by two main metaphors: (1) that of the biological cell in which numerous enzymes are manufactured on demand when certain kinds of processing are needed, and (2) that of courting, marriage, and divorce, according to which structures on all levels have natural "affinities" for one another, and they seek to bond together into larger units, which may further bond into larger units, or which may, when unstable, get dismantled. We are also working on making our operating-system-like architecture capable of "self-watching" and altering its own course according to the results of that self-watching, a process that we call "self-guidance".

Stan C. Kwasny

Interpreting Errorful Inputs in Natural Language Understanding Systems

We are investigating how techniques designed for interpreting syntactically ill-formed inputs in a Natural Language Understanding System can be applied to other stages of language. Specifically addressed will be the question of whether these or other techniques can operate at the semantic and pragmatic levels to achieve a more robust understanding at those levels. When this is achieved, the overall robustness of the language processing component of such a system will be enhanced.

In this study, a Cascaded Augmented Transition Network (CATN) model of grammar is utilized. This permits interaction between several Augmented Transition Network (ATN) grammars written for various stages of language processing. It is our theory that malformations realized at the surface or syntactic level may also occur at other levels as well. Furthermore, the nature of these deeper malformations is identical in many ways to the nature of malformations that have been successfully treated at the syntactic level. This theory will be tested during the course of this work.

John O'Donnell

List Processing Architecture Research

A computer architecture consisting of a binary tree of combinational logic, with data paths connecting adjacent leaves, is being applied to list processing languages (LISP and SCHEME). This architecture supports a data structure which combines the operations performed on lists and arrays, and efficiently implements variable binding access and storage management in LISP. The research studies the requirements of efficient language implementation, the capabilities and limitations of the base architecture, and will develop hardware design techniques to support physical fabrication of the architecture. Physical fabrication of several versions of the architecture will test the design techniques, permit a detailed evaluation of the architecture, and may lead to useful high level language machines.

Franklin Prosser

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 with Professor David Winkel of the University of Wyoming. Our interest in teaching digital design led us to develop hardware instructional laboratories at Indiana University and at the University of Wyoming, and we wrote a current textbook (Winkel and Prosser, *The Art of Digital Design*, Prentice-Hall, Inc.). The labs and the book contribute to my goal of teaching students to deal with complexity in orderly and systematic says; in the lab each student builds a fully functional minicomputer from MSI-level components, using structured design and debugging techniques.

In our research in structured design, we are developing 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 interface, and copious space 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 FORTH machine for high-speed execution of programs written in the FORTH language.

Also, using a simplified version of the Logic Engine, I am revising my hardware instructional laboratory, so that students may produce their minicomputer designs using either hardwired or microprogrammed control elements.

Soon I hope to develop a VLSI version of the Logic Engine control circuits, since the main portion of the Engine should fit nicely into a VLSI design.

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 of control algorithms. These techniques form the backbone of our textbook. Recently, I have made some progress in incorporating these and similar techniques into VLSI design.

Paul W. Purdom, Jr. and Cynthia A. Brown

Average Time Complexity of NP-Complete Problems

The worst case complexity of the best known algorithms for solving NP-complete problems is exponential. We are studying the average time complexity of satisfiability using various search algorithms on models of random conjunctive normal form predicates. A class of predicates over v variables is characterized by $p(v)$, the probability a given literal is in a random clause, and $t(v)$, the number of random independently selected clauses in a predicate. We are characterizing the functions $p(v)$ and $t(v)$ for which some algorithm takes polynomial average time in the limit as v becomes large. We have found algorithms that take polynomial average time when $p(v)$ or $t(v)$ grow extremely slowly and when $t(v)$ grows quickly compared to $p(v)$.

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 best 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.

Compiler Design

The use of attribute grammars to specify compilers is being studied.

Edward L. Robertson

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. Moreover, 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 1) discover properties characterizing NP-complete problems, 2) study when it is possible to find approximate solutions to NP-complete problems given that finding exact solutions is too costly, and 3) develop methods to ignore hopeless cases if a case-by-case search for solutions is indeed necessary.

Mitchell Wand and Daniel P. Friedman

Algebraic and Logical Semantics of Computation

Our research in the semantics of computation has focused on the use of continuations and their representations in a variety of contexts. We are extending this investigation to study implementations and representations of denotational semantics. In particular, we are studying compiler correctness with respect to Scott-Strachey semantics. We provide alternatives to the use of so-called congruence relations for this task. We have developed techniques for deriving target code as a representation of continuation semantics for a sizable class of languages. These techniques have implications for the design of language-oriented machines in microcode or VLSI; we hope to explore these implications as well.

David S. Wise and Daniel P. Friedman

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 language 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 preformance with respect to program maintenance and automatic introduction of sequentality to reduce time and space needs.

§1 Publications

A. Publications Appearing May, 1981–May, 1982.

Brown, C.A. and Purdom, P.W., Jr. "Average Time Analysis of Backtracking," *SIAM J. on Computing* (1981) pp. 583–593.

Brown, C.A. and Purdom, P.W., Jr. "How to Search Efficiently," *Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia (1981) pp. 588–594.

Brown, C.A. and Purdom, P.W., Jr. "Average Time for Satisfiability Algorithms," *Nineteenth Annual Allerton Conference on Communication, Control and Computing*, Monticello, Ill. (1981) p. 644.

Brown, C.A. and Purdom, P.W., Jr. "An Empirical Comparison of Backtracking Algorithms," *IEEE Trans. on Pattern Recog. and Mach. Intell. 4* (1982) pp. 309–316.

Burns, J. "Symmetry in Systems of Asynchronous Processes," *Proc. 22nd Symp. on Foundations of Computer Science*, Oct. 1981.

Burns, J. "Data Requirements for Implementation of N-process Mutual Exclusion using a Single Shared Variable," *Journal of the ACM 29*, 1, Jan. 1982 (with P. Jackson, N.A. Lynch, M.J. Fischer and G.L. Peterson)

Clinger, W. "Foundations of Actor Semantics" (Ph.D. Thesis), MIT Artificial Intelligence Technical Report 633, May 1981.

Epstein, G. "Positive Multiple-valued Switching Functions—An Extension of Dedekind's Problem," *Proc. of 12th Int'l. Symp. on Multiple-valued Logic* (with Y.W. Liu).

Epstein, G. "A summary of 'Core Points in Double Heyting Algebras and Dissectable Lattices'," *Polish Acad. of Sci., Bull. of Section of Logic, 10*, 4 (1981), pp. 181-184, (with A. Horn).

Filman, R.E. and Friedman, D.P. "Models, Languages, and Heuristics for Distributed Computing," *National Computer Conference 1982*, AFIPS Press.

Hofstadter, D.P. "Heisenberg's Uncertainty Principle and the Many-Worlds Interpretation of Quantum Mechanics," *Scientific American*, July, 1981.

Hofstadter, D.P. "Analogies and Roles in Human and Machine Thinking," *Scientific American*, September, 1981.

Hofstadter, D.P. "Nonlinear Iteration and Strange Attractors," *Scientific American*, November, 1981.

Hofstadter, D.P. "Self-Referential Sentences Revisited," *Scientific American*, January, 1982.

Hofstadter, D.P. "The Skeptical Inquirer versus the National Enquirer," *Scientific American*, February, 1982.

Hofstadter, D.P. "Is the Genetic Code Arbitrary?" *Scientific American*, March, 1982.

Hofstadter, D.P. "Pattern, Poetry, and Power in the Music of Frederic Chopin," *Scientific American*, April, 1982.

Hofstadter, D.P. "On Number Numbness," *Scientific American*, May, 1982.

O'Donnell, J. "A Systolic Associative LISP Computer Architecture with Incremental Parallel Storage Management," TR81–5, Computer Science Dept., University of Iowa (1981).

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

Robertson, E.L. "Continual Pattern Replication," *Information and Control 48*, 3, March, 1981, pp. 211–220 (with J. Ian Munro).

Robertson, E.L. "On the Structure of Sets in NP and other Complexity Classes," *Theoretical Computer Science 15* (1981) pp. 181–200 (with L.H. Landweber and R.J. Lipton).

Wand, M. "Semantics-Directed Machine Architecture," *Conf. Rec. 9th ACM Symp. on Principles of Prog. Lang.* (1982), pp. 234–241.

Wand, M. "Specifications, Models, and Implementations of Data Abstractions," *Theoretical Computer Science 20* (1982), pp. 3–32.

Wise, D.S. "Interpreters for Functional Programming," In J. Darlington, P. Henderson, and D.A. Turner (ed), *Functional Programming and its Applications*, Cambridge Univ. Press (1982), pp. 253–280.

B. Articles to Appear

Clinger, W. "Nondeterministic Call by Need is neither Lazy nor by Name," to appear in the *Proceedings of the 1982 ACM Conference on LISP and Functional Programming*, August, 1982.

Clinger, W., Friedman, D.P. and Wand, M. "A Scheme for a Higher-Level Semantic Algebra," *Proc. US-French Seminar on the Application of Algebra to Language Definition and Compilation*, (Fontainebleau, France, June, 1982) (J. Reynolds and M. Nivat, eds.) to be published.

Epstein, G. "Core Points in Double Heyting Algebras," to appear, *Algebra Universalis.*

Epstein, G. "The Underlying Ground for Hypothetical Propositions," to appear, *Scientia.*

Filman, R.E. and Friedman, D.P. *Coordinated Computing: Tools & Techniques for Distributed Software*, to be published by McGraw Hill (1983).

Hofstadter, D.R. "Who Shoves Whom Around Inside the Careenium?" to appear, *Synthese.*

Hofstadter, D.R. "Subcognition and Computation: A Reply to Allen Newell," to appear, *Knowledge*, edited by Fritz Machlup.

Keutzer, K.W. and Robertson, E.L. "The M-Shuffle as an Interconnection Network for SMD Machines," *Twentieth Annual Allerton Conference on Communication, Control and Computing*, Monticello, Ill. (1982).

Kwasny, S. "Ill-Formed and Non-Standard Language Problems," position paper, *20th Annual Meeting of the Association for Computational Linguistics*, 16–18 June, 1982, Toronto, Canada.

Purdom, P.W., Jr. and Brown, C.A. "An Analysis of Backtracking with Search Rearrangement," to appear, *Siam J. Comp.*

Purdom, P.W., Jr. and Brown, C.A. "Evaluating Search Methods Analytically," *National Conf. on Artificial Intelligence* (1982), pp. 124-127.

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

Robertson, E.L. "On the Complexity of Partitioning Sparse Matrix Representations," to appear, *BIT* (with J.P. Malmquist).

Wand, M. "Deriving Target Code as a Representation of Continuation Semantics," *ACM Trans. on Prog. Lang. and Systems 4*, 3 (July, 1982), pp. 496–517.

Wand, M. "Loops in Combinator-Based Compilers," to appear, *Conf. Rec. 10th ACM Symp. on Principles of Prog. Lang.* (1983).

Wand, M. "What is Lisp?" to appear, *American Mathematical Monthly.*

Wise, D.S. "Functional Programming," in A. Ralston (ed.) *Encyclopedia of Computer Science* (1982 revision).

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.

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.

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 114 — Steven D. Johnson. "Connection Networks for Output Driven List Multiprocessing," (October, 1981).

TR 116 — Steven D. Johnson. "Circuits and Systems Implementing Communication with Streams," (October, 1981).

TR 117 — Paul W. Purdom, Jr. "Solving Satisfiability Problems with Less Searching," (October, 1981).

TR 118 — Paul W. Purdom, Jr. and Cynthia A. Brown. "Polynomial Average–Time Satisfiability Problems," (December, 1981).

TR 119 — A. T. Kohlstaedt. "Daisy 1.0 Reference Manual," (November, 1981).

TR 120 — S.D. Johnson and A.T. Kohlstaedt. "DSI Program Description," (November, 1981).

TR 121 — Lee Becker. "Phonological Analysis by Computer: Prospects &Direction," (December, 1981).

TR 122 — Mitchell Wand. "Research in the Computer Science Department at I.U. 1980-81," (December, 1981).

TR 123 — Paul W. Purdom, Jr. "Search Rearrangement, Backtracking, and Polynomial Average Time," (March, 1982).

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," (1982).

TR 128 — Paul W. Purdom, Jr. and Cynthia A. Brown. "The Pure Literal Rule and Polynomial Average-Time," (June, 1982).

TR 129 — L. Edblom and D.P. Friedman. "Issues in Applicative Real-Time Programming," (August, 1982).

§5 Colloquium Speakers in Computer Science 1981-82

Giuseppe Longo
University of Pisa/MIT
Semantics of Lambda–Calculus: An Introductory Account
September 9, 1981

Steve Johnson
Ph.D. Candidate, Indiana University
The 1981 Summer School at Marktoberdorf
September 15, 1981

Cynthia Brown
Indiana University
Average Time for Satisfiability Algorithms
September 22, 1981

John Barnden
Indiana University
The Hyper–Visual Representation of Symbolic Information in Cognition
October 1, 1981

David Wise
Indiana University
Compact Layouts of Banyan/FFT Networks
October 6, 1981

B. Chandrasekaran
Ohio State University
MDX and Related Medical Decision–Making Systems
October 13, 1981

James E. Burns
Indiana University
Symmetry in Asynchronous Systems
October 20, 1981

G. Berry
Ecole National Superieure des Mines de Paris
Programming with Concrete Data Structures and Sequential Algorithms
October 26, 1981

Guy Steele
Carnegie–Mellon University
The Common Lisp Design Effort
October 27, 1981

Ravi Sethi
Bell Laboratories
Semantics Directed Compiler Generation
November 3, 1981

David B. Pisoni
Indiana University
Voice Technology: Some Results of Research on the Perception of Synthetic Speech Produced by Rule with a Text–to–Speech System
November 17, 1981

Daniel Dennett
Tufts University
The Prospect of a Marriage between Cognitive Ethology and AI
December 4, 1981

Dennis Gannon
Purdue University
Data Driven Scheduling for Parallel Computation: Some Experiments and Structures from Numerical Algorithms
December 8, 1981

Jeff Line
Hewlett Packard Corp.
The HP 3000 Machines
January 26, 1982

Joe Stoy
MIT
The Correctness of Programming Languages Implementations
February 11, 1982

Janice Cuny
Purdue University
Conversion from Data Flow to Synchronous Execution in Loop Programs
February 26, 1982

Jieh Hsiang
University of Illinois
Mechanical Theorem Proving using Term Rewriting Systems
March 4, 1982

Larry Wittie
SUNY Buffalo
Portable Operating Systems for Network Computers
March 17, 1982

Mark Kahrs
University of Rochester
VLSI Compilation for Very High Level Languages
March 18, 1982

Diane Fischer
SUNY Buffalo
Some Results in Applied Statistics
March 29, 1982

Stuart Goldkind
University of Rochester
Introduction to Chess Programming
March 30, 1982

Michael G. Dyer
Yale University
BORIS: An Experiment in In–Depth Understanding Narratives
April 1, 1982

Margot Flowers
Yale University
Reasoning and Memory: A Computer Model of Human Reasoning
April 2, 1982

Eitan Gurari
SUNY Buffalo
Decidable Problems for Programs
April 19, 1982

Carl Smith
Purdue University
A Recursive Talk on Recursion
April 20, 1982