Readings In Scheme

Ozan S. Yigit

York University
Computing and Communication Services

oz@nexus.yorku.ca

Recent Additions to the Scheme Bibliography

[11791] IEEE Std 1178-1990, *IEEE Standard for the Scheme Programming Language*, Institute of Electrical and Electronic Engineers, Inc., New York, NY, 1991.

[Bec91] Brian Beckman, A Scheme for Little Languages in Interactive Graphics, *Software-Practice and Experience 21*, 2 (Feb 1991), 187-207, John Wiley & Sons, Ltd.

Abstract: Programming environments for interactive graphics software typically have a multiplicity of tools applications. Many of these programs contain *ad hoc* ''little language'' interpreters that do many similar things in needlessly different ways. In particular, many little languages have, in addition to their special-purpose constructs, vestigial support for ordinary programming, such as variables, loops and conditionals. If a single, standard programming language were the basis of all these little languages, they could have complete, coherent programming semantics; they could communicate with each other more easily; no design work for basic constructs would be needed; and interpreter implementation work would be saved. The approach of reusing and extending the same core language and interpreter for a variety of little languages is the *extension language* approach.

Scheme is proposed as a good choice for such a core language. Scheme is a simple, elegant, high-level programming language. Extendable implementations are readily available in C source form. Example applications in Scheme from interactive graphics are presented that would be nearly impossible to code in a typical scripting language and very tedious to code in a lower-level implementation language such as C.

[Sif91] Dorai Sitaram and Matthias Felleisen, Modeling Continuations Without Continuations, *Proceedings of the Eighteenth ACM Symposium on Principles of Programming Languages*, 1991, 185-196.

[Teo90] Dan Teodosiu, HARE: A Compiler for Scheme, Master's Thesis, Bucharest Polytechnic Institute, June 1990.

[Teo91] Dan Teodosiu, HARE: An Optimizing Portable Compiler for Scheme, *ACM Sigplan Notices 26*, 1 (Jan 1991).

Abstract: A highly optimizing Scheme compiler called HARE is presented. A combination of several optimization techniques allows for the generation of very efficient code. Easy portability of the compiler has been achieved through the use of a virtual machine as a target for code generation. The compiler will be used as a test-bed for fine-tuning the instruction set of a symbolic architecture, the S-Machine.

A Selection of Abstracts from the Scheme Bibliography

[Bon90] Pierre Bonzon, A Metacircular Evaluator for a Logical Extension of Scheme, *Lisp and Symbolic Computation: An International Journal 3*, 2 (March 1990), 113-133, Kluwer Academic Publishers.

Abstract: We define a computational model for a logical extension of Scheme and give a metacircular evaluator for it. This minimal extension incorporates two new features only, i.e. logical variables and clause expressions, which can be used to define predicates in exactly the same way as lambda expressions can be used to define functions.

Higher-order properties of Scheme are preserved: predicates can be passed to and returned from function applications. Predicate applications can appear as terms in functions. On the other hand, function applications can appear as terms in predicates, and can be formal as well as actual arguments, but only as long as they can be evaluated according to the usual Scheme semantics prohibiting access to unbound variables (except for constructor applications).

[CHO88] William D. Clinger, Anne H. Hartheimer and Eric M. Ost, Implementation Strategies for Continuations, *Conference Record of the 1988 ACM Conference on Lisp and Functional Programming*, August 1988, 124 131.

Abstract: Scheme and Smalltalk continuations may have unlimited extent. This means that a purely stack-based implementation of continuations, as suffices in most languages, is inadequate. Several implementation strategies have been described in the literature. Determining which is best requires knowledge of kinds of programs that will commonly run.

Danvy, for example, has conjectured that continuation captures occur in clusters. That is, the same continuation, once captured, is likely to be captured again. As evidence, Danvy cited the use of continuation in a research setting. We report that Danvy's conjecture is somewhat true in the commercial setting of MacScheme+Toolsmith™, which provides tools for developing Macintosh user interfaces in Scheme. These include an interrupt-driven event system and multitasking, both implemented by liberal use of continuations.

We describe several implementation strategies for continuations and compare four of them using benchmarks. We conclude that the most popular strategy may have a slight edge when continuations are not used at all, but that other strategies perform better when continuations are used and Danvy's conjecture holds.

[FeF86] Matthias Felleisen and Daniel P. Friedman, A Closer Look At Export and Import Statements, *Journal of Computer Languages 11*, 1 (1986), 29-37, Pergamon Press.

Abstract: Export and import statements can be implemented as syntactic extensions. We first define their intuitive semantics in terms of Scheme programs. Then we show how export and import can be improved to allow for arbitrary load-sequence of modules and to handle dynamic extensions of modules.

[FrF90] John Franco and Daniel P. Friedman, Towards A Facility for Lexically Scoped, Dynamic Mutual Recursion in Scheme, *Journal of Computer Languages 15*, 1 (1990), 55-64, Pergamon Press.

Abstract: We propose a facility which allows unbounded associative structures, which we call Dynamic Mutually Recursive Structures (DMRS), in Scheme. An important application is the creation of unbounded vectors and arrays. Another application is as the underpinnings of a global, dynamic letrec capability. A third application is the construction of memo-functions.

Under this facility, DMRS elements are allocated space individually and not until they are side-effected. Thus, a DMRS can be sparse and waste little memory. In addition, the proposed facility removes some of the burdens of writing procedural specifications that are not relevant to functional specifications such as vector boundedness.

[KoW87] Eugene E. Kohlbecker and Mitchell Wand, Macro-by-Example: Deriving Syntactic Transformations from their Specifications, *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages*, Munich, West Germany, Jan 1987, 77-84.

Abstract: This paper presents two new developments. First, it describes a ''macro-by-example'' specification language for syntactic abstractions in Lisp and related languages. This specification language allows a more declarative specification of macros than conventional macro facilities do by giving a better treatment of iteration and mapping constructs. Second, it gives a formal semantics for the language and a derivation of a compiler from the semantics. This derivation is a practical application of semantics-directed compiler development methodology.

[MiR91] James Miller and Guillermo Rozas, Free Variables and First-Class Environments, *Lisp and Symbolic Computation: An International Journal 3*, 4 (1991), 107-141, Kluwer Academic Publishers.

Abstract: A simple set of extensions to the SCHEME language removes the need for a distinguished top level interaction environment by providing first-class environments. These extensions also provide a powerful mechanism for code packaging and may be used to implement simple object-oriented systems. In addition, a mechanism is presented that implements compiled references to free variables as efficiently as in languages like C, provided the code does not directly manipulate first-class environments. The mechanism requires a simple static analysis performed by the compiler and meshes with a slower mechanism used by both interpreted code and compiled code that manipulates first-class environments.

Availability

The complete Scheme bibliography may be obtained in machine-readable *bib* [*refer*] or (automagically-generated) *BibTeX* format via e-mail from oz@nexus.yorku.ca. Both formats are ftp-able from the *Scheme Repository* (currently located at nexus.yorku.ca [130.63.9.66]), under pub/scheme/bib.

Happy scheming! ... oz

Call for Papers

The Twentieth Annual ACM SIGPLAN–SIGACT Symposium on

Principles Of Programming Languages

Charleston, South Carolina, January 11–13, 1993

The twentieth symposium on Principles of Programming Languages will provide a forum for discussion of principles, innovations, and accomplishments in the design, definition, analysis, and implementation of programming languages and systems. Reports on experiences with the application or use of such principles and innovations are encouraged. Papers presented at the symposium must describe work that has not previously been published or presented at a conference.

Many seminal papers have been presented at POPL conferences in the past 20 years. In an attempt to preserve and strengthen that tradition and to maintain broad coverage of the field, the program committee welcomes papers on a diversity of topics, particularly those that set out new directions, and is prepared to increase the number of accepted papers to accommodate them. The symposium is not limited to topics discussed in previous symposia nor to formal approaches. Authors concerned about the appropriateness of a topic may communicate by electronic mail with the program chair prior to submission.

Authors should submit 13 copies (printed double-sided if possible) of a technical summary of a prospective paper to the program chair. The length of the summary must not exceed 5000 words excluding bibliography and figures. **Excessively long summaries will be rejected immediately by the program chair.**

The summary should explain the contribution of the paper, both in general and in technical terms. It is important to identify what has been accomplished, to explain why it is significant, and to compare with previous work. Papers will be judged on originality, significance, correctness, and clarity. Authors should make every effort to make the technical content of their papers understandable to a broad audience.

Submissions must be received by **July 27, 1992**. They should include a return postal address and an electronic mail address (if available). Authors will be notified of the acceptance or rejection of their papers by **September 14, 1992**. Full versions of the accepted papers must be received in camera-ready form by **October 20, 1992**. Authors of accepted papers will be required to sign ACM copyright release forms. Proceedings will be distributed at the conference and subsequently will be available for purchase from ACM Press.

Program Chair
Susan L. Graham
Computer Science Division – EECS
University of California
Berkeley, CA 94720 USA
graham@cs.berkeley.edu
(510) 642-2059

General Chairs
Mary Van Deusen
IBM Watson Res. Lab
maida@watson.ibm.com
Bernard Lang
INRIA Rocquencourt
lang@margaux.inria.fr

Local Arrangements Chair
Dee Medley
Augusta College
dmedley@uscn.bitnet
(404) 737-1672

Program Committee:

Martin Abadi · *DEC SRC*
Hans Boehm · *Xerox PARC*
Charles Consel · *Yale Univ.*
Ron Cytron · *IBM Watson Res. Lab*
Susan L. Graham · *Univ. Calif. Berkeley*
Gilles Kahn · *INRIA Sophia Antipolis*

Monica Lam · *Stanford Univ.*
James Larus · *Univ. Wisconsin*
Gary Lindstrom · *Univ. Utah*
David McQueen · *AT&T Bell Labs*
Frank Pfenning · *Carnegie Mellon Univ.*
Guy Steele · *Thinking Machines Corp.*

SIGPLAN PROCEEDINGS

EASY TO ORDER!

POPL - 19th Annual Symposium on Principles of Programmming Languages
Albuquerque, NM, January 19 - 22, 1992. Sponsored by ACM SIGACT and SIGPLAN. 376 pages, ISBN: 0-89791-453-8 • Order No. **549920**, Nonmembers: $53.00, ACM Members: $26.00

POPL - Annual Symposium on Principles of Programmming Languages
Orlando, FL, January 21-23, 1991. Sponsored by ACM SIGACT and SIGPLAN. 366 pages, ISBN: 0-89791-419-8 • Order No. **549910**, Nonmembers: $27.00,ACM Members: $22.00

POPL - 17th Annual Symposium on Principles of Programmming Languages
San Francisco, CA, January 17 - 19, 1990 . Sponsored by ACM SIGACT and SIGPLAN. 402 pages, ISBN: 0-89791-343-4 • Order No. **549900** Nonmembers: $36.00, ACM Members: $24.00

POPL - 16th Annual Symposium on Principles of Programmming Languages
Austin, TX, January 11 - 13, 1989. Sponsored by ACM SIGACT and SIGPLAN. 352 pages, ISBN: 0-89791-294-2 • Order No. **549890**, Nonmembers: $27.00, ACM Members $21.00

OOPSLA '91
Phoenix, AZ, October 6- 11, 1991. Sponsored by ACM SIGPLAN. 384 pages, ISBN: 0-201-55417-8 • Order No. **548911**, Nonmembers: $35.50*, ACM Members: $20.00

OOPSLA /European Conference on OOP'90
Ottawa, Ontario, October 21- 25, 1990. Sponsored by ACM SIGPLAN. 336pages, ISBN: 0-201-52430-X • Order No. **548901**, Nonmembers: $35.50*, ACM Members: $20.00

OOPSLA '89
New Orleans, LA, October 1- 6, 1989. Sponsored by ACM SIGPLAN. 528 pages, ISBN: 0-201-52249-7 • Order No. **548893**, Nonmembers: $35.50*, ACM Members: $28.00

OOPSLA '88
Sponsored by ACM SIGPLAN. Order No. **548881** Nonmembers: $38.00, ACM Members: $26.00

OOPSLA '87
Sponsored by ACM SIGPLAN. Order No. **548871** Nonmembers: $48.00, ACM Members: $33.00

PPOPP '91 -3rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programmming
Williamsburg, Virginia, April 21- 24, 1991. Sponsored by ACM SIGPLAN. ISBN: 0-89791-390-6 • Order No. **551910**, Nonmembers: $18.00, ACM Members: $13.00

PPOPP '90 - 2nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programmming
Seattle, Washington, March 14- 16, 1990. Sponsored by ACM SIGPLAN. 206 pages. ISBN: 0-89791-350-7 • Order No. **551900**, Nonmembers: $25.00, ACM Members: $19.00

ASPLOS -IV 4th International Conference on Architectural Support for Programming Languages and Operating Systems
Santa Clara, CA, April 8- 11, 1991. Sponsored by ACM SIGPLAN, SIGARCH and SIGOPS. • Order No. **556910**, Nonmembers: $25.00, ACM Members: $20.00

ASPLOS -III 3 rd International Conference on Architectural Support for Programming Languages and Operating Systems
Boston, MA, April 3- 6, 1989. Sponsored by ACM SIGPLAN, SIGARCH and SIGOPS. 303 pages, ISBN : 0-89791-300-0 • Order , NO **556890**, Nonmembers: $29.00, ACM Members: $20.00

SIGPLAN '91- 4th Conference on Programming Language, Design and Implementation
Sponsored by ACM SIGPLAN. • Order No. **548910**, Nonmembers: $27.00, ACM Members: $20.00

SIGPLAN '90 - 3rd Conference on Programming Language, Design and Implementation
White Plains, NY, June 20-22, 1990. Sponsored by ACM SIGPLAN. 358 pages. ISBN: 0-89791-364-7 • Order No. **548900**, Nonmembers: $31.00, ACM Members: $22.00

SIGPLAN '89 - 2 nd Conference on Programming Language, Design and Implementation
Portland, OR, June 21-23, 1989. Sponsored by ACM SIGPLAN. 355 pages. ISBN: 0-89791-306-X • Order No. **548892**, Nonmembers: $32.00, ACM Members: $22.00

SIGPLAN '88 - 1st Conference on Programming Language, Design and Implementation
Sponsored by ACM SIGPLAN. • Order No. **548880**, Nonmembers: $35.00, ACM Members: $24.00

FPCA '89 - Proceedings of Functional Programming Languages, and Computer Architecture
London, UK, September 11-13, 1989. Sponsored by ACM SIGPLAN and IFIP. 396 pages. ISBN: 0-201-51389-7• Order No. 407890, Nonmembers: $35.50*, ACM Members: $28.00

Proceedings of the 3rd Software Engineering Symposium on Practical Software Development Environments
Boston, MA, November 28-30, 1988. Sponsored by ACM SIGPLAN and SIGSOFT. 268 pages. ISBN: 0-89791-290-X, • Order No 594880, Nonmembers: $28.00, ACM Members: $20.00

Workshop on Object-Based Concurrent Programming
San Diego, CA, September 26-27, 1989. Sponsored by ACM SIGPLAN. ISBN: 0-89791-304-3, • Order No 548891, Nonmembers: $22.00, ACM Members: $15.00

PPEALS '88 - ACM SIGPLAN Symposium on Parallel Programming Experience with Applications, Languages, and Systems
New Haven, CT, July 19-21, 1988. Sponsored by ACM SIGPLAN. 246 pages. ISBN: 0-89791-276-4, Order No. 551880, Nonmembers: $26.00, ACM Members: $19.00

Proceedings of the ACM Workshop on Parallel and Distributed Debugging
1991. Sponsored by ACM SIGPLAN and SIGOPS. • Order No 548912, Nonmembers: $19.00, ACM Members: $15.00

Proceedings of the ACM W orkshop on P arallel and Distributed Debugging
Madison , WI, May 5-6, 1988. Sponsored by ACM SIGPLAN and SIGOPS. 404 pages. ISBN: 0-89791-296-9, • Order No 548884, Nonmembers: $29.00, ACM Members: $20.00

Common Lisp Object System Specification: American National Standards Institute (ANSI) Standard Document for Editorial Review
1988. Sponsored by ACM SIGPLAN. ISBN: 0-89791-289-6, • Order No 548883, Nonmembers: $20.00, ACM Members: $13.00

1990 Symposium on LISP and Functional Programming
Nice, France, June 25-29, 1990. Sponsored by ACM SIGPLAN, SIGACTand SIGART. 348 pages. ISBN: 0-89791-368-X, Order No 552880, Nonmembers: $31.00, ACM Members: $24.00

1988 Symposium on LISP and Functional Programming
Snowbird, UT, July 25-27, 1988. Sponsored by ACM SIGPLAN, SIGACTand SIGART. 364 pages. ISBN: 0-89791-273-X, • Order NO 552880, Nonmembers: $36.00, ACM Members: $24.00

EASY TO ORDER
Telephone Orders (Credit Cards Only)

CALL TOLL FREE 1-800-342-6626
For Customer Service Call: 301-528-4261
In AK, MD, and Outside USA: 301-528-4261
Please Have your Credit Card Number Handy

Proceedings may be ordered prepaid from
ACM Order Department
P.O. Box 64145, Baltimore, MD 21264

* **Special ordering note:** Nonmembers must order OOPSLA proceedings after 1988 and FPCA '89 from Addison-Wesley Publishing Co., Order Dept., Jacob Way, Reading, MA 01867. Members can order their copies from ACM.

EDITORIAL POLICY

All submissions to Lisp Pointers, with the exception of technical articles, should be made in camera-ready text and sent to the appropriate department head. Technical articles may be submitted to the Technical Articles Editor in either hard copy or in TEX source files by Arpanet link, tar format cartridge tape, or tar format reel-to-reel. All submissions should be single-spaced with no page numbers. Without a special waiver from the appropriate department head, submissions will be limited to ten pages. This can be achieved by printing longer articles two-up. Camera-ready text is defined to be no more than 7 1/2 x 10 inches or 19 x 25 centimeters, centered on an 8 1/2 x 11 inch page. Articles that contain too much blank space will be rejected. It is the author's responsibility to retain a working copy of the submission, as contributions will not be returned to authors. Authors not fluent in writing English are requested to have their work reviewed and corrected for style and syntax prior to submission.

Although Lisp Pointers is not refereed, acceptance is subject to the discretion of the appropriate department head. The scope of topics for Lisp Pointers includes all dialects of Lisp and Scheme. We encourage research articles, tutorials, and summarizations of discussions in other forums. Lisp Pointers is not a forum for detailed discussions on proposed changes to the Common Lisp standard.

Lisp Pointers is a Special Interest Publication of the Special Interest Group on Programming Languages (SIGPLAN). A subscription to LISP Pointers does not include membership in any group.

Note: ACM Members who expect to renew their membership within the next six months should send no LISP Pointers subscription payment now as they will be billed on their renewal notice. Those expecting to renew in seven or more months should send on half the annual LISP Pointers subscription payment now.

____
Name (Please print or type)

____
Mailing Address

____

____
City State Zip

Signature______________________________

New address. Please change my ACM record.

Annual subscription rates are
$12 for ACM Members,
$7 for ACM Student Members,
$25 for Non-ACM Members.

ACM MEMBER
ACM Member No. _________
(see note regarding dues payment)

ACM STUDENT MEMBER
ACM Student Member No. _____
(see note regarding dues payment)

NON-ACM MEMBER
Enclosed is annual subscription
payment of $25

Please send information on ACM Membership.

Please make checks payable to ACM Inc. and mail to: ACM, Inc., P.O. Box 12115,
Church Street Station, New York, N.Y. 10249.