Readings In Scheme

Ozan S. Yigit

York University
Computing and Communication Services

oz@nexus.yorku.ca

Intro Blurb

In this edition of the *readings*, we have the entries from the special Scheme issue of the BIGRE† journal and for the first time, the abstracts for a random selection of entries. Future editions of this column will continue to feature such abstracts.

A machine readable copy of this bibliography is available via e-mail in *bib* [*refer*] format, and there is also an automagically-generated *BibTeX* version.

Happy scheming.

A Selection of Abstracts

[Fel88] Matthias Felleisen, λ-vs-CS: An Extended λ-Calculus for Scheme, *Conference Record of the 1988 ACM Conference on Lisp and Functional Programming*, August 1988, 72-85.

Abstract: The λ-v-CS-calculus is a conservative extension of the λ-value-calculus for reasoning about programs in Scheme-like languages. The basis of the extended calculus is a symbolic rewriting semantics for imperative programs. We show with numerous examples how to state and prove equational properties of Scheme-programs in the calculus. The examples suggest that the algebraic manipulation of imperative-functional programs is as feasible and as fruitful as that of functional ones.

[Ste77] Guy Lewis Steele Jr., Debunking the "Expensive Procedure Call" Myth, or Procedure Call Implementations Considered Harmful, or LAMBDA, the Ultimate GOTO, *ACM Conference Proceedings*, 1977, 153-162.

Abstract: Folklore states that GOTO statements are "cheap", while procedure calls are "expensive". This myth is largely a result of poorly designed language implementations. The historical growth of this myth is considered. Both theoretical ideas and an existing implementation are discussed which debunk this myth. It is shown that the unrestricted use of procedure calls permits great stylistic freedom. In particular, any flowchart can be written as a "structured" program without introducing extra variables. The difficulty with the GOTO statement and the procedure call is characterized as a conflict between abstract programming concepts and concrete language constructs.

[CuR90] Pavel Curtis and James Rauen, A Module System for Scheme, *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, Nice, France, June 1990.

---

† BIGRE Bulletin, *Putting Scheme to Work*, André Pic, Michel Briand, Jean Bézivin, (Eds.), No. 65, July 1989

Abstract: This paper presents a module system designed for large-scale programming in Scheme. The module system seperates specification of objects from their implementations, permitting the seperate development, compilation and testing of modules. The module system also includes a robust macro facility.

We discuss our design goals, the design of the module system, implementation issues and our future plans.

[JrS78] Guy Lewis Steele Jr. and Gerald Jay Sussman, The Art of the Interpreter, or the Modularity Complex (parts zero, one, and two), MIT AI Memo 453, Massachusetts Institute of Technology, Cambridge, Mass., May 1978.

Abstract: We examine the effects of various language design decisions on the programming styles available to a user of the language, with particular emphasis on the ability to incrementally construct modular systems. At each step we exhibit an interactive meta-circular interpreter for the language under consideration. Each new interpreter is the result of an incremental change to a previous interpreter.

We explore the consequences of various variable binding disciplines and the introduction of side effects. We find that dynamic scoping is unsuitable for constructing procedural abstractions, but has another role as an agent of modularity, being a structured form of side effect. More general side effects are also found to be necessary to promote modular style. We find that the notion of side effect and the notion of equaltiy (object identity) are mutually constraining; to define one is to define the other.

The interpreters we exhibit are all written in a simple dialect of LISP, and all implement LISP-like languages. A subset of these interpreters constitute a partial historical reconstruction of the actual evolution of LISP.

[FFJ90] John Franco, Daniel Friedman and Steven Johnson, Multi-way Streams in Scheme, *Journal of Computer Languages 15*, 2 (1990), 109-125.

Abstract: We present a mechanism for the maintenance of streams based on the Scheme facility of call-with-current-continuation or call/cc. The mechanism supports stream sharing and has overheadcost which is independent of top-level program parameters if call/cc is implemented in heap-based systems. It is shown how the control structure of call/cc can save programming effort in cases where multiple procedures output to the same stream in irregular order.

[DFF87] Bruce F. Duba, Matthias Felleisen and Daniel P. Friedman, Dynamic Identifiers can be Neat, Computer Science Technical Report No. 220, Indiana University, Bloomington, Indiana, April 1987.

Abstract: Linguistic facilities to express dynamic extent are common in modern, lexically-scoped Lisps. In such languages there is the traditional possibility of having identifiers with *dynamic scope*, or the alternative of having statically bound identifiers with *dynamic extent*. The latter possibility is superior since it inherits all the advantages of lexical scoping, however, the known implementation techniques interfere with the correct optimization of tail-recursion. We propose a new implementation strategy and show that it does not suffer from this problem.

Additions to the Bibliography

[AsS89] J. Michael Ashley and Richard M. Salter, A Revised State Space Model for a Logic Programming Embedding in Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Dan89a] Olivier Danvy, Combiner Logiquement en Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Dan89b] Olivier Danvy, Programming with Tighter Control, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Del89] Vincent Delacour, Picolo Expresso, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[DDC89] Alain Deutsch, Renaud Dumeur, Charles Consel and Jean-Daniel Fekete, CSKIM: An Extended Dialect of Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[DFF87] Bruce F. Duba, Matthias Felleisen and Daniel P. Friedman, Dynamic Identifiers can be Neat, Computer Science Technical Report No. 220, Indiana University, Bloomington, Indiana, April 1987.

[Fel88] Matthias Felleisen, λ-vs-CS: An Extended λ-Calculus for Scheme, *Conference Record of the 1988 ACM Conference on Lisp and Functional Programming*, August 1988, 72-85.

[KaL89] Simon M. Kaplan and Joseph P. Loyall, GARP/Scheme: Implementing a Concurrent, Object-Based Language, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Kim89] Tan Gon Kim, The DEVS-Scheme Modelling and Simulation Environment, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[LaF89a] Guy Lapalme and Marc Feeley, Micro-Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[LaF89b] Julia L. Lawall and Daniel P. Friedman, Embedding the Self Language in Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[PiB89] André Pic and Michel Briand, Visual Programming with Generators, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Que89] Christian Queinnec, Validation Suite Generation, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[RBC89] J. C. Royer, J. P. Braquelaire, P. Casteran, M. Desainte-Catherine and J. G. Penaud, Le modèle OBJScheme: principes et applications, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Sé89] Nitsan Séniak, Compilation de Scheme par spécialisation explicite, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Str89] Robert Strandh, OOOZ, A multi-User Programming Environment Based on Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

[Ulr89] John Wade Ulrich, Enumeration Algorithms and Non-deterministic Programming in Scheme, *BIGRE Bulletin (Putting Scheme to Work)*, 65, July 1989.

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
$8 for ACM Members,
$4 for ACM Student Members,
$18 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 $18

☐ **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.