standard-output

Scheme Standardization

Christopher T. Haynes[1]
Chair, IEEE/MSC/P1178 Working Group on Scheme

This is a brief report of efforts to standardize the Scheme programming language. Scheme inherits Lisp's rich set of symbol manipulation primitives, latent storage allocation, dynamic type checking. and simple syntax. Scheme is distinguished from most Lisp dialects by a single variable environment, block structure with static scope, and uniform evaluation of the operator and operand positions of a procedure call. Since there is no storage penalty for tail-recursive procedure calls, they may be used to express iteration. Provision is made for a rich set of numerical types, and exact and inexact numbers are distinguished. The ability to create first-class escape procedures allows almost all known forms of sequential control to be expressed. Above all, Scheme achieves its expressive power through the simplicity and generality of its design, and not by the accumulation of features. (The draft standard is about 50 pages long.) The reader may wish to consult books by Abelson and Sussman [1], Springer and Friedman [2], and Dybvig [3], among others, for tutorial introductions to Scheme.

Scheme was designed in 1975 by Gerald Sussman and Guy Steele as "a simple concrete experimental domain for certain issues of programming semantics and style." [4] In 1978, Sussman and Steele published "The Revised Report on Scheme: A Dialect of Lisp," [5] reflecting early evolution in the design of Scheme. By 1984, the use of Scheme in research, teaching, and system development had spread from MIT to several other universities and industrial laboratories. This was accompanied by a proliferation of Scheme implementations, developed in a spirit of innovation. To enhance code portability and consistency in the use of Scheme as a publication language, a workshop was held at Brandeis University in 1984. Participants included representatives for the Scheme implementations of MIT, Yale, Indiana University, and Texas Instruments. The consensus reached at this workshop, with further refinements through committee work and network discussions, was recorded in "The Revised Revised Report on Scheme" [6].

[1]Computer Science Department, Indiana University, Bloomington, IN 47405, 812/855-3376, haynes@cs.indiana.edu

Further revisions, reflecting consensus reached largely through networked discussions, resulted in "The Revised^3 Report on the Algorithmic Language Scheme," which appeared in *SIGPLAN Notices* in December, 1986. [7] The authors of this series of reports on Scheme, known as the R^nRS, met again at MIT in the summer of 1987 and at the 1988 ACM Conference on Lisp and Functional Programming in Snowbird, Utah, where agreement was reached on further changes to be incorporated in a subsequent R^4RS. Publication of this report is expected in the near future.

In March, 1988, a majority of the R^nRS authors met as a study group at Indiana University with representatives of Institute of Electrical and Electronics Engineers (IEEE) and X3 standardization committees to consider initiation of a formal effort to standardize Scheme. It was concluded that a formal standard was desirable to improve code portability, publication uniformity, and language visibility. It was further recommended that this effort be initiated through the IEEE Microprocessor Standards Committee (MSC), which subsequently approved formation of a Working Group on Scheme with project authorization P1178. The MSC was chosen because it had experience with language standardization, and the IEEE's procedures for standardization working groups were less formal than those of X3 and oriented toward the participation of individuals, rather than organizations. Furthermore, IEEE standardization procedures are approved by the American National Standards Institute (ANSI), allowing many IEEE standards to become ANSI standards. (Smolin [8] reviews IEEE standardization procedures.)

The study group felt strongly that the R^nRS authors should continue their language design work and publish further reports. While these reports are likely to form the basis for future revisions of the standard, they may be less conservative than is appropriate for a standard. They may, for example, include experimental features that should be withheld from standardization. The study group considered it essential that the publication of an IEEE standard based on the R^nRS, and copyrighted by the IEEE, not compromise the public domain status of the R^nRS. The IEEE indicated that this would not be a problem. Finally, the study group concluded that Scheme standardization would not conflict with standardization of other members of the Lisp family, such as X3J13's development of a Common Lisp standard and work on a Lisp standard by the international standards organization ISO.

The Working Group on Scheme has held four meetings, on July 27, 1988

(Snowbird), February 3, 1989 (MIT), July 7, 1989 (MIT), and January 19, 1990 (San Francisco). These meetings have been supplemented by extensive network discussions and subcommittee meetings. At the last meeting the draft standard was approved for submission to the MSC for ballotting. The fourth version of the draft standard (P1178/D4) is now in the ballotting process. If all goes smoothly, the draft could be approved by the IEEE Standards Board in September and published by the end of this year. Following approval of the standard, the Working Group will dissolve. The $R^n RS$ authors will, however, continue their work of extending and refining the specification of Scheme, and an IEEE Working Group on Scheme will be reconstituted to revise the standard within at most five years. (It is possible that the present draft will be approved only as a "trial-use" standard, in which case revision would be required within two years.)

The draft standard for Scheme is based in large part on a draft of the $R^4 RS$, and it is expected that an implementation conforming to the draft standard will also conform to the $R^4 RS$. (Conformance cannot be certain now because the $R^4 RS$ is not complete at the time of this writing.)

From the $R^3 RS$ to the draft standard for Scheme there have been a large number of minor improvements, both in the language defined and in its specification. Perhaps the most significant addition is a distinction between exact and inexact numbers that is unique to Scheme, and should enhance the robustness of numeric programs. [9] The distinction between essential and inessential features in $R^3 RS$ has been eliminated by making some of the inessential features essential and dropping others. The empty list is now interpreted as a true value, and must thus be distinct from the false value, `#f`.

Though most Scheme implementations support a syntactic extension (macro) mechanism, the draft standard does not include a syntactic extension facility. The Scheme community has been reluctant to standardize on a facility that does not satisfactorily address the variable capture problem that has plagued traditional macros. A solution appears to have been found, and is likely to be published as an appendix of the $R^4 RS$. If this or a related mechanism proves to be satisfactory in practice, it may be incorporated in a revision of the Scheme standard.

Reluctance to prematurely standardize a syntactic extension mechanism is characteristic of the Scheme standardization effort. It is intended that the Scheme standard encourage continued language design in a variety of areas, such as packaging and object-oriented programming facilities and interfaces for users and operating systems.

The rapid progress of the Working Group on Scheme was made possible by the high quality of the R^4RS, which reflects several years of effort by its authors. Special acknowledgment is due the R^nRS editors, William Clinger and Jonathan Rees, and the draft standard editors, Chris Hanson, James Miller, and David Bartley.

References

[1] Harold Abelson and Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, 1985.

[2] George Springer and Daniel P. Friedman. *Scheme and the Art of Programming.* MIT Press, 1989.

[3] R. Kent Dybvig. *The Scheme Programming Language.* Prentice-Hall, 1987.

[4] Gerald Jay Sussman and Guy Lewis Steele Jr. *Scheme: An Interpreter for Extended Lambda Calculus.* MIT Artificial Intelligence Memo 349, December 1975.

[5] Guy Lewis Steele Jr. and Gerald Jay Sussman. *The Revised Report on Scheme, a Dialect of Lisp.* MIT Artificial Intelligence Memo 452, January 1978.

[6] William Clinger, editor. *The Revised Revised Report on Scheme; or, An Uncommon Lisp.* MIT Artificial Intelligence Memo 848, August 1985, and Computer Science Department Technical Report 174, Indiana University, June 1985.

[7] Jonathan Rees and William Clinger, editors. The revised^3 report on the algorithmic language Scheme. *ACM SIGPLAN Notices* 21(12), pp. 37–79, December 1986.

[8] Michael Smolin. MicroStandards. *IEEE MICRO*, August and October, 1987.

[9] William Clinger. The Scheme of things: exact and inexact numbers. *LISP Pointers* 2(1), pp. 46–49, July–September 1988.