Draft Standard for the Scheme Programming Language

***** P1178/D4 — March 7, 1990 *****

Foreword

(This Foreward is not a part of P1178/D4, Draft Standard for the Scheme Programming Language.)

Programming languages should be designed not by piling feature on top of feature, but by removing the weaknesses and restrictions that make additional features appear necessary. Scheme demonstrates that a very small number of rules for forming expressions, with few restrictions on how they are composed, suffice to form a practical and efficient programming language that is flexible enough to support most of the major programming paradigms in use today.

Scheme places few restrictions on the use of procedural abstractions: procedures are full first-class objects. Although Scheme is a block-structured language, and it permits side-effects, it differs from most imperative block-structured languages by encouraging a functional style of programming that uses procedures to encapsulate state.

In a similar spirit, Scheme implementations impose no storage penalty for tail-recursive procedure calls, and continuations (which are present, although behind the scenes, in all programming languages) are first-class Scheme objects that act like procedures. This permits nearly all known sequential control structures to be expressed in terms of procedure calls.

Purpose of this standard

Throughout its thirty-year life, the Lisp family of languages has continually evolved to encompass changing ideas about programming-language design. Scheme has participated in the evolution of Lisp. Scheme was one of the first programming languages to incorporate first-class procedures as in the lambda calculus, thereby proving the usefulness of static scope rules and block structure in a dynamically typed language. Scheme was the first major dialect of Lisp to distinguish procedures from lambda expressions and symbols, to use a single lexical environment for all variables, and to evaluate the operator position of a procedure call in the same way as an operand position. Scheme was the first widely used programming language to rely entirely on procedure calls to express iteration and to embrace first-class escape procedures.

Specifying a Standard for Scheme is intended to encourage the continued evolution of Lisp dialects by identifying a coherent set of constructs that is large enough to support the implementation of substantial programs, but also small enough to admit significant extensions and alternate approaches to language design. For example, this Standard does not mandate the inclusion in Scheme of large run-time libraries, particular user interfaces, or complex interactions with external operating systems, although practical Scheme implementations ordinarily provide such features.

In particular, there are important linguistic design issues that are not discussed in this Standard *precisely because* Scheme has sparked fruitful new approaches in these areas, and this Working Group does not wish to discourage the further development of good ideas by taking a position on issues under active investigation. Some of these issues are macros, packaging, and object-oriented programming.

The Working Group hopes that future revisions of this Standard will be sensitive to the fact that good ideas need time to mature, and that exploration can often be stifled by the premature adoption of standards.

Background

The first description of Scheme was written in 1975 [?]. A revised report [?] appeared in 1978, which described the evolution of the language as its MIT implementation was upgraded to support an innovative compiler [?]. Three distinct projects began in 1981 and 1982 to use variants of Scheme for courses at MIT, Yale, and Indiana University [?, ?, ?]. An introductory computer science textbook using Scheme was published in 1984 [?].

As Scheme became more widespread, local dialects began to diverge until students and researchers occasionally found it difficult to understand code written at other sites. Fifteen representatives of the major implementations of Scheme therefore met in October 1984 to work toward a better and more widely accepted Standard for Scheme. Their report [?] was published at MIT and Indiana University in the summer of 1985. Subsequent rounds of revision took place in the spring of 1986 [?], and at a meeting that preceded the 1988 ACM Conference on Lisp and Functional Programming [**?**]. The Working Group for this Standard first met at that same conference; the Standard draws heavily on the earlier reports.

Members of the Scheme Working Group of the Microprocessor Standards Subcommittee and those who participated by correspondence were as follows:

Christopher T. Haynes, *Chairman*
David H. Bartley, **Chris Hanson**, **James S. Miller**, *Editors*

Harold Abelson	Norman Adams	Cyril N. Alberga	Joel Bartlett
Scott Burson	Clyde Camp	Bill Campbell	Jerome Chailloux
Stewart Clamen	William Clinger	Pavel Curtis	Jeffrey Dalton
Olivier Danvy	Klaus Dassler	Kenneth Dickey	Bruce Duba
R. Kent Dybvig	Marc Feeley	Daniel P. Friedman	Mark Friedman
Richard P. Gabriel	John Gateley	Arthur Gleckler	Patrick Greussay
Jed Harris	Robert Hieb	Takayasu Ito	Roger Kirchner
Paul Kristoff	Tim McNerney	William Maddox	Sidney Marshall
Robert Mathis	Richard Mlynarik	Andy Norman	Eric Ost
John D. Ramsdell	Jonathan Rees	Guillermo J. Rozas	Benjamin Schreiber
George Springer	Guy L. Steele Jr.	Gerald Jay Sussman	Eric Tiedemann
Queyen Tran	R. L. Tritchard	Mitchell Wand	Jon L White
Taiichi Yuasa			

CONTENTS