The R6RS Status Report

Marc Feeley
Université de Montréal
feeley@iro.umontreal.ca

Editors' note: This article is the lightly edited text of the progress report submitted by the Scheme Language Editors Committee to the Scheme Language Steering Committee on September 2, 2004. We have included it in the workshop proceedings to represent the concluding presentation of the workshop on the state of the standardisation effort by the editors committee.

The members of the Scheme Language Editors Committee are:
Marc Feeley, editor in chief (Université de Montréal)
Will Clinger (Northeastern University)
Kent Dybvig (Indiana University)
Matthew Flatt (University of Utah)
Richard Kelsey (Ember Corporation)
Manuel Serrano (INRIA)
Michael Sperber (DeinProgramm)

The members of the Scheme Language Steering Committee are:
Alan Bawden (Brandeis University)
Guy L. Steele Jr. (Sun Microsystems)
Mitch Wand (Northeastern University)

–Waddell & Shivers

At the 2003 Scheme workshop in November, the strategy committee (Alan Bawden, Will Clinger, Kent Dybvig, Matthew Flatt, Richard Kelsey, Manuel Serrano, Mike Sperber) was given a mandate to nominate a steering committee and an editors committee to work on the R6RS standard. In January 2004, the editors committee was nominated: Feeley (editor in chief), Clinger, Dybvig, Flatt, Kelsey, Serrano, and Sperber.

On January 19, a private mailing list was created to keep a record of the email exchanges between the editors. Although some editors suggested that a more open process would be desirable, we chose to keep this mailing list private to avoid outside interference and keep the process disciplined and focused. Sometime in the future the archive of the discussions will be made public so that the reasons for the design decisions are clear.

Because of the expected difficulty in managing productive discussions for a seven-member committee by email, we adopted some ground rules for ensuring progress. If an editor does not participate in an email discussion within a reasonable time limit (which was set to seven days), then the other editors may assume that editor does not have an opinion on the subject (or does not want to voice his opinion), and can be ignored (in the discussion, in a vote, *etc.*). I think this has been helpful to create a certain pressure to keep up-to-date in the discussion.

The subject of backward compatibility was also discussed early on. That is, will R5RS code work unchanged in an R6RS-compliant Scheme implementation? Our position is that backward compatibility is desirable but that there may be some incompatibilities (for example at the lexical syntax level) that prevent R5RS code from working under R6RS. Our first objective is to improve the Scheme language. Backward compatibility, while important, is a secondary objective.

We then set off on our first technical task: come up with a list of goals that is more precise than the one in the draft charter (which had four items: produce a core Scheme specification, define a module system, define a macro system, and designate library modules). Our plan was to use this list of goals (1) to organize the design process, and (2) to identify which changes were uncontroversial (and thus easier to standardize) and which would require considerable effort (and where consensus might not be achievable during the R6RS design process).

All the editors were polled to get a list of specific issues that they thought needed to be addressed in the R6RS design process (*i.e.*, features the committee should consider adding/removing). At the end of March, we had merged all the editors' lists into a single list with each editor's position. At this point, there had been very little technical discussion of these issues (on purpose), so that we could order the issues and discuss them in a disciplined way. As suggested by Will Clinger, the list was organized into the following categories:

- Deletions of R5RS-Scheme features;
- Incompatible changes to R5RS Scheme;
- Extensions that could be entirely compatible with R5RS Scheme

Permission to make digital or hard copies, to republish, to post on servers or to redistribute to lists all or part of this work is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To otherwise copy or redistribute requires prior specific permission.
Fifth Workshop on Scheme and Functional Programming. September 22, 2004, Snowbird, Utah, USA. Copyright 2004 Marc Feeley.

  - but would break some implementation-specific extensions;
  - but would be controversial and aren't worth it;
  - that are controversial or difficult but necessary;
  - that are probably uncontroversial.

Below is the list of issues, without each editor's position. Note that this list is still open to be expanded as new issues arise in the design process.

Deletions from R5RS

- remove `transcript-on` and `transcript-off`
- remove `force` and `delay`
- remove multiple values

Incompatible changes to R5RS

- make syntax case-sensitive

Extensions that would break implementation-specific features

- specify evaluation order
- support for processes
- support for network programming
- object-oriented programming
- external representation for records
- serialization

Extensions to R5RS (controversial and probably unnecessary)

- pattern matching / destructuring
- abstract data type for continuations
- composable continuations
- box types
- uninterned symbols
- extended symbol syntax
- add `letrec*`, define internal `define` in terms of it
- optional and keyword arguments as in DSSSL

Extensions to R5RS (controversial or difficult but necessary)

- module system
- non-hygienic macros
- records
- mechanism for new primitive types
- Unicode support
- errors and exceptions
- require a mode where "it is an error" means "an error is signaled"

Extensions to R5RS (probably not terribly controversial)

- multiline comments
- external representation for circular structures
- `#!eof`
- more escape characters
- require that `#f`, `#t`, and characters be followed by a delimiter
- `case-lambda`
- `cond-expand`
- allow the name of the macro being defined in `syntax-rules` to be arbitrary (or _)
- allow continuations created by `begin` to accept any number of values
- tighten up specification of `eq?` and `eqv?` (or otherwise address their portability problems)
- tighten up specification of inexact arithmetic
- add `+0`, `-0`, `+inf`, `-inf`, `+nan`
- bitwise operations on exact integers
- SRFI 4 homogeneous numeric vectors
- specify dynamic environment
- operations on files
- binary I/O or new I/O subsystem entirely
- string code
- regular expressions
- command-line parsing
- hash tables
- library for dates
- system operations

Editorial changes

- split language into core and libraries

Additional extensions

- expression comments
- subset of Common Lisp `format` (in a library)

Because of the central role of the module system and its probable use in splitting the Scheme language into a core and libraries, we decided that the most pressing issue was the design of the module system. Our starting point was the "strawman module system" proposed by Flatt, which is based on the MzScheme system. Various aspects of the proposed system were discussed, mainly to understand it better and to add constructive criticism. Because many aspects are interrelated, we did not achieve consensus on any specific aspect (nor did we really try to achieve it given that this is early in the design process).

Over May and June, the discussion on the module system was slow and only two of the seven editors were active. At the end of June, I suggested that the reason for this apathy might be a lack of practical experience with the proposed module system (the two editors that were active both had experience with the MzScheme module system). I proposed that we should work on building a portable implementation of the module system so that the editors can all experiment with it in our own Scheme implementations. This would get the editors more involved in the details of the module system, allow proposed changes to be made and evaluated on-the-fly by changing the portable implementation, and the resulting public-domain code would greatly increase acceptance of R6RS by other implementors. It still remains to be seen if this portable implementation becomes a reality, as it represents quite a bit of work.

Dybvig noted that there are few differences between the module system proposed by Flatt and the one in Chez Scheme. This prompted an effort by Dybvig and Flatt to design a new module system that combines both systems. There has been a very active discussion since then.

We have made arrangements to have a whole-day meeting (September 18) in Snowbird to discuss these issues face to face. All editors will be there, except for Kelsey. We expect the module system to be the main topic of discussion and to make significant progress. We will also start discussing other issues on our list.