R5RS: Legacy Standard Language

Version 4.2.3

December 1, 2009

The The Revised⁵ Report on the Algorithmic Language Scheme defines a dialect of Scheme. We use R^5RS to refer to both the standard and the language defined by the standard.

The default dialect of Scheme provided by mzscheme and other PLT Scheme tools differs from R⁵RS in many ways, but PLT Scheme includes tools and libraries for running R⁵RS programs.

See §21 "Dialects of Scheme" for general information about different dialects of Scheme within PLT Scheme.

Contents

1	Running R ⁵ RS Programs	3
2	plt-r5rs	4
3	R ⁵ RS Module Language	5
	3.1 Non-R ⁵ RS Bindings from r5rs	5
	3.2 Notes on R ⁵ RS Functions	5
4	R ⁵ RS Initialization Library	7

1 Running R⁵RS Programs

PLT Scheme provides several layers of support for programs written according to R⁵RS:

- DrScheme provides an R5RS language, which can be selected via the Language|Choose Language... menu item. See §1.2 "Choosing a Language" in *DrScheme: PLT Programming Environment* for more information.
- The plt-r5rs executable runs an R⁵RS program or provides a read-eval-print loop for evaluating R⁵RS expressions and definitions. See §2 "plt-r5rs" (later in this manual) for more information.
- The r5rs library implemented R⁵RS procedures and syntactic forms. It can also be used with #lang to create a module whose body is implemented in an R⁵RS-like language. See §3 "R⁵RS Module Language" (later in this manual) for more information.
- The r5rs/init library extends r5rs to set parameters (such as case-insensitive symbol reading) for R⁵RS loading or an R⁵RS read-eval-print loop. See §4 "R⁵RS Initialization Library" (later in this manual) for more information.

2 plt-r5rs

The plt-r5rs executable runs an R⁵RS program from a file that is supplied on the command line. If no program file is provided as a command-line argument, then a read-eval-print loop is started.

Before starting a read-eval-print loop, an initialization file is loaded, if it exists. The file is the same as the file reported by (find-system-path 'init-file), but with the characters mzscheme in the filename replaced by pltr5rs. For example, under Unix, the file is "~/.pltr5rsrc".

By default, plt-r5rs departs from R⁵RS conformance in one crucial way: the names of pre-defined functions cannot be redefined at the top level. This restriction enables better runtime performance. Use the -no-prim command-line flag—before a file to load, if any—to obtain the standard behavior for primitive bindings (at the cost of performance).

3 R⁵RS Module Language

```
#lang r5rs
```

As a library, r5rs provides the syntactic forms and procedures defined by R⁵RS. When used as a language via #lang, the program is read with the following parameterizations:

```
(read-case-sensitive #f)
(read-accept-infix-dot #f)
(read-curly-brace-as-paren #f)
(read-square-bracket-as-paren #f)
```

The r5rs bindings can be imported into a top-level environment, and then evaluation in that top-level environment corresponds to R⁵RS. Use (namespace-require/copy 'r5rs) with an empty namespace to maximize conformance with R⁵RS; Using (namespace-require 'r5rs), in contrast, creates primitive bindings as imports, which is the same as using plt-r5rs without the -no-prim flag. More simply, use (scheme-reportenvironment 5). See also r5rs/init, which sets reader and printer parameters to increase conformance.

Using r5rs via #lang creates a module whose body is implemented with an R⁵RS-like language. The main difference from R⁵RS is that, as a module language, r5rs does not allow redefinition of top-level bindings, and expressions evaluated through load and eval cannot automatically access bindings defined within the module.

3.1 Non-R⁵RS Bindings from r5rs

In addition to the bindings defined by R⁵RS, the r⁵rs library provides the following bindings from mzscheme (which are not legal identifiers in R⁵RS syntax, so there is no danger of collisions in R⁵RS programs):

```
#%app #%datum #%top #%top-interaction #%require #%provide
```

It also provides mzscheme's #%plain-module-begin as #%module-begin. Note that #%require can be used to import PLT Scheme libraries into an otherwise R⁵RS program, and #%provide can be used to export from a module that is implemented in an R⁵RS-like language.

3.2 Notes on R⁵RS Functions

The cons of r5rs corresponds to scheme/base's mcons. Similarly, cdr is mcdr, and map is scheme/mpair's mmap, and so on.

An R⁵RS *environment* is implemented as a scheme/base *namespace*. Also, relative to scheme/base, the expr passed to eval is constructed using mutable pairs.

The scheme-report-environment function returns a namespace containing the bindings of r5rs. Procedure values are installed into the namespace using namespace-require/copy, so that they can be redefined.

The null-environment function returns a namespace containing the syntactic forms of r5rs, not including #%module-begin (which is not useful outside of a module).

4 R⁵RS Initialization Library

```
(require r5rs/init)
```

The r5rs/init module re-exports r5rs, and also sets parameters as follows:

```
(read-case-sensitive #f)
(read-accept-infix-dot #f)
(read-curly-brace-as-paren #f)
(read-square-bracket-as-paren #f)
(print-mpair-curly-braces #f)
```

The side-effect of setting these parameters is useful when the module is required before loading an R⁵RS program, so that the reader and printer behave more as specified in R⁵RS. In particular, the plt-r5rs executable initializes by importing r5rs/init.