R^{5}RS Errata

Richard Kelsey

September 29, 1999

Several inconsistencies in R^{5}RS have been pointed out since its initial publication on the World-Wide Web. This document addresses the most important of them.

Semantics of `call-with-current-continuation`

The definition of `call-with-current-continuation` in section 7.2 is incorrect because it is incompatible with `dynamic-wind`. As shown in section 4 of [1], however, this incorrect semantics is adequate to define the `shift` and `reset` operators, which can then be used to define the correct semantics of both `dynamic-wind` and `call-with-current-continuation`.

Redefinition of keywords

As stated in section 1.2 keywords are not reserved. The grammar specified in section 7.1 is thus somewhat misleading because a program may redefine any or all syntactic keywords. In particular, it is possible to use any identifier as a variable.

`Port?`

Section 3.2 includes `port?` in the list of predicates defining disjoint types; there is no such procedure. `Input-port?` and `output-port?` should be included in this list of predicates. An object may satisfy both `input-port?` and `output-port?`, but cannot satisfy any other pair of predicates in this list.

`Begin` at top level

As with `define`, there are two different semantics for `begin` depending on context. When used as an expression a `begin` form contains expressions and is library syntax as described in section 4.2. At top level a `begin` form contains commands and definitions and its semantics are as described in section 5.1: a top-level `begin` is equivalent to the sequence of commands and definitions it contains. A top-level `begin` is syntax, not library syntax.

A `begin` at top level that contains only expressions is syntactically ambiguous, but the two semantics are equivalent in that particular case.

`List-ref`

The description of (`list-ref` *list* k) in section 3.2.2 should say

> It is an error if *list* has k or fewer elements.

and not

> It is an error if *list* has fewer than k elements.

Reference

[1] Andrzej Filinski. Representing Monads. In *Proceedings of the 1994 ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 446–457.