Subject: Addressing comments in 02-209 concerning 02-167r1
From: Van Snyder

1 Edits

Edits refer to 02-007r2. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[The word "overriding" should almost surely be "generic".] 45:21-23

C459 (R442) A generic binding shall have a passed-object dummy argument (4.5.1.6) if and only if all generic bindings with the same *generic-spec*, both inherited (4.5.3.1) and declared within a type declaration, have a passed-object dummy argument.

2 An independent proposition

In reviewing whether a draft of this paper answered the Editor's objections to 02-167r1, he remarked that he liked the introduction in 02-167r1 better than its edits for 4.5.1.6. The discussion presently in 4.5.1.6 doesn't address procedure pointer components with implicit interface. Here is a revised 4.5.1.6:

The **passed-object dummy argument** is a distinguished dummy argument specified in the interface of a type-bound procedure or a procedure pointer component. It affects procedure overriding (4.5.3.2) and argument association (12.4.1.1). 51:8-18

C463$\frac{1}{3}$ The passed-object dummy argument shall be a scalar, nonpointer, nonallocatable dummy argument with the same declared type as the type being defined.

C463$\frac{2}{3}$ The passed-object dummy argument shall be polymorphic if and only if the type being defined is extensible.

[Notice that 02-219 makes a change similar to part of this sentence at [51:11].]

If NOPASS is specified, there is no passed-object dummy argument.

If the interface is implicit or has no dummy argument that meets the requirements specified by constraint C463$\frac{1}{3}$, there is no passed-object dummy argument. It is permitted to confirm this by specifying NOPASS.

If PASS is not specified and the first dummy argument does not meet the requirements specified by constraint C463$\frac{1}{3}$, there is no passed-object dummy argument. It is permitted to confirm this by specifying NOPASS.

If neither PASS nor NOPASS is specified and the first dummy argument meets the requirements specified by constraint C463$\frac{1}{3}$, it is the passed-object dummy argument.

If PASS is specified the first argument is the passed-object dummy argument.

If PASS(*arg-name*) is specified the dummy argument named *arg-name* is the passed-object dummy argument.

[The "polymorphic" part is now required by constraint C463$\frac{2}{3}$, so we don't need to specify it again. Editor: Delete "The … extensible."] 43:33

[The "polymorphic" part is now required by constraint C463$\frac{2}{3}$, so we don't need to specify it again. Editor: Delete "That ... extensible."] 44:2-3

[The "polymorphic" part is now required by constraint C463$\frac{2}{3}$, so we don't need to specify it again. Editor: Delete "It ... extensible."] 45:13-14

[The "polymorphic" part is now required by constraint C463$\frac{2}{3}$, so we don't need to specify it again. Editor: Delete "That ... extensible."] 45:18

[The last four edits are also specified in 02-219.]

3 Not sure what to do

Concerning [56:14], the Editor asks "Shall correspond to what? And by what definition? By name, position, either, both?"

This begs the question "Do we need 'correspond' by any definition?"

If not:

---

(5) Arguments at corresponding positions in the reduced dummy argument lists (12.4.1) shall have the same names and characteristics. 56:14-16

(6) Except for their types, the passed-object dummy arguments shall have the same characteristics.