Date: 2 November 1998
To: J3
From: Van Snyder
Subject: Simplify PASS_OBJ (by getting rid of it)

1 Background

In 98-007r3, the discussion of the `PASS_OBJ` attribute of a type-bound procedure, and its effect on argument association, is a convoluted mess. I know, because I wrote it. It could probably be written better, but I think it would be better to substitute something easier to implement, easier to describe, and therefore easier to teach and use.

2 Proposal

Delete the `PASS_OBJ` attribute from type-bound procedure declarations, and procedure pointer component declarations.

Add a declaration to subroutine and function headers, say `SELF (` *pass-obj* `)`, that denotes the "dummy argument" with which the "invoking object" is associated. Example:

```
TYPE :: T
...
  PROCEDURE P => Q
END TYPE T
...
SUBROUTINE Q ( ARG ) SELF ( PASS_OBJ )
...
END SUBROUTINE Q
...
  TYPE(T) :: X
  ...
  CALL X%P ( A ) ! X%P is bound to Q, X is associated with Q's
    ! SELF ''argument'' PASS_OBJ, and A is associated with Q's ARG.
```

Malcolm proposed this in correspondence several months ago, but neither of us liked it. It didn't appear to allow `Q` to be used on its own, or with several different types. If, however, we allow the argument named in the `SELF` declaration to be `OPTIONAL`, then `Q` can be used on its own. If the passed-object in 98-007r3 isn't optional, there's no real reason to call the procedure without the `X%` above – doing so only re-arranges the correspondence between actual and dummy arguments. If it is, you get the same result both ways.

The unlikely case of using a procedure with several types could be accomplished using type-bound procedures that contain a procedure invocation without the `X%`.

3 Edits

Edits refer to 98-007r3. 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 immediately following text, while a page and line number followed by +

indicates that immediately following text is to be inserted after the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

These edits illustrate that this isn't a massive change; it mostly amounts to deleting text.

[Editor: delete] [40:43]

[Editor: delete] [41:7-9]

R438 *binding-attr* **is** NON_OVERRIDABLE [41:29-30]

[Editor: delete] [41:32-34]

Add "The interface shall specify a SELF variable that has the type *type-name*" to the constraint. [41:38]

```
PROCEDURE LENGTH => POINT_LENGTH
```
[47:19]

```
REAL FUNCTION POINT_LENGTH ( B ) SELF ( A )
```
[47:21]

If a binding overrides an inherited one then the two type-bound procedures or abstract interfaces shall match in the following ways: [52:31-33]

[Editor: Remove "Except for the passed-object dummy argument," and capitalize "the". [53:1]

```
PROCEDURE LENGTH => POINT_3D_LENGTH
```
[53:12]

```
REAL FUNCTION POINT_3D_LENGTH ( B ) SELF ( A )
```
[53:14]

[Editor: Add "whether it has a SELF argument" to the list.] [230:6]

Should we allow procedure references of the form `X%Y` where `Y` is a specific or generic procedure name, `Y` is not a component of the type of `X`, and `Y` has a SELF argument of the type of `X`? [238:9,16] *J3 note*

[Editor: delete] [240:16-28]

If a procedure is referenced by the *data-ref* % *binding-name* form, and *binding-name* is the binding name of a procedure that has a SELF argument, then the *data-ref* is an actual argument, and it is associated with the dummy argument of the procedure specified by SELF. [240:29+]

If a procedure is referenced by the *variable* form, *variable* consists of `derived_type_variable % procedure_pointer_component`, and `procedure_pointer_component` has an interface that includes a SELF variable, then the `derived_type_variable` is an actual argument, and it is associated with the dummy argument of the procedure specified by SELF.

■ ([*dummy-arg-name-list*]) [*special-args*] [248:16]

R1217a *special-args* **is** RESULT (*result-name*) [SELF (*self-name*)]
or SELF (*self-name*) [RESULT (*result-name*)]

Constraint: The *self-name* shall be a scalar nonpointer of derived type.

[Editor: After the note] [249:22+]
If SELF is specified then *self-name* is a dummy argument. It is associated with an actual argument as specified in 12.4.1.2.

■ [([*dummy-arg-list*])] [SELF (*self-name*)] [250:4]

Constraint: The *self-name* shall be a scalar nonpointer of derived type.

If SELF is specified then *self-name* is a dummy argument. It is associated with an actual argument as specified in 12.4.1.2. [250:22+]

[Editor: Replace by] ■ [*special-args*] [250:44]

Constraint: The *self-name* shall be a scalar nonpointer of derived type. [251:10]

If SELF is specified then *self-name* is a dummy argument. It is associated with an actual argument as specified in 12.4.1.2. [251:27+]

[Editor: Delete] [365:16-30]