Subject: Thoughts stimulated by issue 216
From: Van Snyder

1 Introduction

In issue 216, the editor suggests "The interaction between ASYNCHRONOUS or VOLATILE and the ASSOCIATE or SELECT TYPE construct needs to be described."

In pondering this, I began also to wonder how the ALLOCATABLE, INTENT, OPTIONAL, POINTER and TARGET attributes of the [type] selector should be related to the associate name. In the case of the ASSOCIATE construct, but not the SELECT TYPE construct, it is specified that the associate name does not have the ALLOCATABLE or POINTER attribute.

A macro substitution model would work for the ASSOCIATE construct, by saying something like "the effect is as if the *associate-name* were replaced by the *selector* everywhere the *associate-name* appears within the block." It's more problematical in the case of the SELECT TYPE construct because of the changing type of the *associate-name*. I think it's too much of a mess to make it work, just to allow a [type] selector with a vector subscript.

In any case, the rules about attributes for *associate-name* should be the same in the two constructs.

The intent during the design was to make construct association similar to argument association. The similarity was not exact, however, in that the type, type parameters, "polymorphicity," rank and bounds "leak through" the association without being specified. Allowing the ALLOCATABLE, ASYNCHRONOUS, INTENT, OPTIONAL, POINTER, TARGET and VOLATILE attributes to "leak through" as well has useful functionality. E.g. it allows one to deallocate an otherwise non-deallocatable result of a pointer function by way of an associate name, or to allocate, deallocate or pointer assign a component that has numerous complicated *part-refs*. In my opinion it is better to have an exact correspondence to argument association, which would require declaring the associate name(s) within the block, or to let the other seven relevant attributes noted above "leak through" the association also. It also means that *all* of the properties of the *selector* "leak through," not five out of twelve.

The following edits allow the above six attributes to "leak through" the association. In particular, the section on construct association (14.6.1.4) is expanded to describe the association completely. Since it is referenced in the descriptions of both the SELECT TYPE and ASSOCIATE constructs, the rules are the same, except for the type and polymorphicity, which are described in the separate sections.

2 Edits

Edits refer to 00-007. 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.

[Editor: Replace "*type-selector*" by "*selector*" and replace "type selector" by "selector" throughout. Yes, I've checked – a global search-and-replace, if FRAME offers such a feature, is OK.] 152:19-153:34

Constraint: If the *selector* shall not appear in a variable definition context (14.7.7), the *associate-name* shall not appear in a variable definition context. 152:29-30

[Editor Replace "type, type parameters, rank, and bounds" by "type parameters." "Type" is incorrect, and the "rank and bounds" are specified in 14.6.1.4.] 153:10

Constraint: If the *selector* shall not appear in a variable definition context (14.7.7), the *associate-name* shall not appear in a variable definition context. 154:35-36

[Editor: Replace ", type parameters, rank, and bounds" by "and type parameters." The "rank and bounds" are specified in 14.6.1.4.] 154:47

[Editor: Delete "If the selector" to the end of the paragraph.] 155:1-4

[Editor: Delete note 216.] 155:5-8

Execution of a SELECT TYPE statement establishes an association between the selector and the associate name of the construct. Execution of an ASSOCIATE statement establishes an association between each selector and the corresponding associate name of the construct. 351:43-352:5

The associate name has the same rank and bounds as the selector.

The associate name has the ALLOCATABLE, ASYNCHRONOUS, INTENT, OPTIONAL, POINTER, TARGET or VOLATILE attribute if and only if the selector has the attribute.

If the selector is a *variable* other than an array with an array section having a vector subscript, the associate name is associated to the data object and is definable if and only if the *variable* is definable. Otherwise if the associate name does not have the POINTER attribute it has the value of the selector, which is evaluated prior to execution of the block, and it is not definable.

R1401 *associate-name* **is** *name*

Constraint: If the *associate-name* has the ALLOCATABLE or POINTER attribute, and the INTENT(IN) attribute, it shall not be allocated or deallocated.

Constraint: If the *associate-name* has the POINTER attribute and the INTENT(IN) attribute, it shall not be a *pointer-object* in a nullify statement (6.3.2) or pointer assignment statement (7.5.2).

If the *associate-name* has the POINTER attribute and does not have the INTENT(IN) attribute, it can be deallocated if the selector can be deallocated or if the selector is a function result that can be deallocated.

Note 14.14$\frac{1}{2}$

If the selector is a pointer to an array section that is not a whole array, to a non-pointer component, or to an allocatable object it cannot be deallocated.