Subject: Numerous changes in 12.4.1.2, hopefully including resolution of Issue 79
From: Van Snyder

1 Introduction

Issue 79 implicitly addresses numerous issues. The treatment of the interaction of pointer association status, allocation status and intent, and the treatment of polymorphism and type parameters, are fractured, dispersed and incomplete, although they might be completed elsewhere. I do not propose to re-write the section, but only to make a small dent in correcting, completing and reorganizing it.

The discussion of INTENT(OUT) is actually a bit of over-kill, because a conspiracy of 5.1.2.3 and 6.3 covers it all.

2 Edits

Edits refer to 99-007r2. 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 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: Delete from "That is" to the end of the paragraph.] 77:22-24

The sentence repeats material at the immediately previously cited places. It is incomplete concerning type parameters, and making it complete would make it even more redundant.	*Note to J3*

[Editor: Add a new paragraph. Identical text is proposed in 99-185.] 82:2+

Deferred parameters of an undefined or disassociated pointer are undefined [new section] (14.8). The definition status of deferred parameters of a pointer object is the same as the definition status of the corresponding parameters of the object with which it is associated. If the deferred parameters of the object with which the pointer is associated are defined, the values of those parameters of the pointer object are the same as the values of the object with which it is associated.

The above is intended to cover both pointer assignment and argument association.	*Note to J3*

[Editor: Replace from "If a dummy argument..." to the end of the paragraph with a new paragraph:] 275:6-9

If a dummy argument or its associated actual argument, or both, are polymorphic, and the dummy argument has INTENT(IN) or is neither allocatable nor a pointer, the declared type of the actual argument shall be an extension type (4.5.3) of the declared type of the dummy argument. If the dummy argument is polymorphic, is allocatable or a pointer, and has INTENT(OUT), the actual argument shall be polymorphic, and the declared type of the dummy argument shall be an extension type of the declared type of the actual argument. If the dummy argument is allocatable or a pointer and has neither INTENT(IN) nor INTENT(OUT), the declared type of the actual argument shall be the same as the declared type of the dummy argument.

The above rules guarantee that if the dummy argument is allocated or pointer associated within the procedure, the dynamic type of the associated actual argument becomes an extension type of its declared type.	Note 12.21½

If the dummy argument is polymorphic, does not have INTENT(OUT), and the associated actual argument is defined, then upon invocation of the procedure the dummy argument assumes the dynamic type of the associated actual argument. If it has INTENT(OUT) or the associated actual argument is undefined, the initial dynamic type of the dummy argument is undefined.

[Editor: Replace "agree with" by "have the same value as".] 275:10

If a dummy argument has assumed type parameters, the initial definition status of these parameters is assumed from corresponding parameters of the actual argument; if the corresponding parameters of the actual argument are defined, the values are assumed also. 275:20-42

The following is needed, in spite of the addition at 82:2+, to cover allocatable dummy arguments	*Note to J3*

If a dummy argument does not have INTENT(OUT), the same process applies to its deferred parameters, if any. The initial values of deferred parameters of dummy arguments with INTENT(OUT) are undefined. When execution of the procedure completes, if the dummy argument does not have INTENT(IN), the definition status of each parameter of the actual argument that corresponds to a deferred parameter of the dummy argument becomes that of the corresponding parameter of the dummy argument; if a deferred parameter of the dummy argument is defined, the value of the corresponding parameter of the actual argument becomes that of the parameter of the dummy argument.

At the invocation of the procedure, an allocatable dummy argument becomes deallocated if it has INTENT(OUT). If it does not have INTENT(OUT) then it receives the allocation status of the actual argument and, if the actual argument is allocated, the dummy argument becomes associated with the same object. The allocation status of an allocatable dummy argument may change during the execution of the procedure unless it has INTENT(IN). When execution of the procedure completes, the allocation status of the actual argument becomes the allocation status of the dummy argument unless the dummy argument has INTENT(IN). 277:14+

Compare to existing discussion of pointer dummy arguments at 276:10-18.	*Note to J3*

[Editor: Add a new section 14.8. Identical text is proposed in 99-185 and 99-187.] 379:10+

14.8 Definition and undefinition of deferred type parameters

A deferred type parameter of an object may be defined or may be undefined and its definition status may change during execution of a program. An action that causes a deferred type parameter to become undefined does not imply that it was previously defined. An action that causes a deferred type parameter to become defined does not imply that it was previously undefined.

The definition status of deferred type parameters is changed by the following events:

1. Successful execution of an ALLOCATE statement or allocation of an allocatable component during intrinsic assignment (7.5.1.5) causes deferred type parameters of the allocated object to become defined.

2. Pointer assignment causes the values of deferred parameters of *pointer-object* and its ultimate components to assume the definition status of corresponding parameters of *target* and its ultimate components.

3. Events that cause a pointer to become undefined (14.6.2.1.3) or to become disassociated

(14.6.2.1.2) cause the deferred type parameters of the pointer to become undefined.

4. Deallocating (6.4.3) a pointer or an allocatable object causes the deferred type parameters of the object and its subobjects to become undefined. This includes deallocation of an allocatable component of a derived type object during intrinsic assignment (7.5.1.5).

5. Reference to a procedure causes the deferred type parameters of dummy arguments that do not have INTENT(OUT), and their subobjects, to assume the same definition status as corresponding type parameters of corresponding actual arguments and their subobjects.

6. Any change in the definition status of deferred type parameters of an object causes the same change in the corresponding deferred type parameters of an associated object of the same type.

If the change suggested for VALUE at 85:26 in 99-133 is accepted, add "other than a dummy arugment that has the VALUE attribute" after the first "object" in the last item above.	*Note to J3*