Subject: Comments on Section 7
From: Van Snyder

1 Edits

Edits refer to 01-007r3. 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 (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[There is no *scalar-mask* term. Editor: "*scalar-mask*" ⇒ "*scalar-mask-expr*".] 139:19

[Editor: "max" ⇒ "*max*" twice.] 141:41,43

[The term *target* has been split into *data-target* and *proc-target*. The term *data-target* is the right one here. Editor: "*target*" ⇒ "*data-target*" twice.] 142:19,22

2 Edits stimulated by 01-283r2

[Sentence "If the pointer ... assignment statement" says that pointers, as such (as opposed to their targets), are primaries. They're not. Editor: Delete "as a primary".] 115:9-12

[The sentence isn't true for functions that have polymorphic results, and whose arguments are functions that have polymorphic results. Insert "declared" before "type".] 119:37-38

[In item 26, 01-283r2 asks "7.5.2.2 [second normative paragraph] In the last two lines one of the *proc-target* terms should be *proc-pointer-object*. I think the second one. Right now it says *proc-target* may be elemental even if *proc-target* isn't." The conclusion was to defer consideration of this. The speculation is correct. Editor: At [135:42] "*proc-target*" ⇒ "*proc-pointer-object*".] 135:40-42

3 Remarks stimulated by 01-283r2

In item 19, 01-283r2 asks "Should the first line say dynamic and the last declared? Doesn't that mix things?" The conclusion was "defer". 130:33-36

This sentence is correct. Don't change it.

In item 20, 01-283r2 asks "The last sentence defining derived type intrinsic assignment appears to be restating (1) and (2) above, but the words are somewhat different. For example is 'accessible defined assignment' in (2) different from 'no accessible generic interface ....' For example, (2) in 7.3.2 goes into detail about where an interface may come from and its typeboundness. Should that apply here? Or does 'not polymorphic' cover it?" 7.5.1.2-3

01-283r2 goes on to ask in item 21 "Do we need to distinguish details about where the interface is at ala (2) in 7.3.2?"

The words should be parallel. The distinction of intrinsic and defined assignment is difficult to explain because intrinsic assignment is defined in terms of "not defined assignment." Defined assignment ought to be described as carefully as defined opeations are, and defined first. Then it becomes easier to define intrinsic assignment as not defined assignment.

In item 22, 01-283r2 asks "the middle clause about nonpointer, nonallocatable. Does this mean that derived type assignment that is NOT type bound to the host type isn't used for 132:36-41

the innards? Suppose I have a type PERSON with a component of type KIDS and with a subroutine that meets the requirements for KID_1 = KID_2. If I do
`PERSON_1 = PERSON_2`
does the generic subroutine ever get invoked? Does it depend on whether or not there is a type-bound assignment interface/routine in PERSON? Or in KIDS? I'm asking about an old fashioned generic assignment and how that fits in with the new type bound stuff. Same comment for the last couple of lines in (2) on the next page. It reads to me that we require 'type bound procedures' when I thought generic assignment could work."

The conclusion was to defer consideration of this.

This is the subject of an interpretation request and a work item.

In item 23, 01-283r2 asks "7.5.1.6 (2) Says there can be a type bound assignment procedure in EITHER X1 or X2. What happens if there is one in both and they are different?" The conclusion was to defer consideration of this. 133:30-33

This can't happen and still satisfy 16.1.2.3.

In item 24, 01-283r2 asks "7.5.2 R736 Does the second line allow something like A%B%pointer => P or only one %? Same question for R740." The conclusion was to defer consideration of this. 134:18-19

This is OK because the syntax exposes the last *part-ref* to the constraints.

In item 25, 01-283r2 asks "7.5.2.1 Second paragraph. Does this mean a non-polymorphic pointer can point to a polymorphic object if the types are OK? Or should dynamic type be declared type." The conclusion was to defer consideration of this. 135:9-10

Yes, but maybe it shouldn't. If "dynamic type" is changed to "declared type" one could still get at the desired data by using a SELECT TYPE construct.

In item 27, 01-283r2 asks "C728. I read this to say that an elsewhere in a named WHERE does not have to be named, but if it is, the name must be the same as the WHERE. It seems odd to require the endwhere to be named if the where is, but not the elsewhere." The conclusion was to defer consideration of this. 137:18-23

This should maybe be changed, but changing it would be incompatible with Fortran 95.