Subject: A few small problems with procedure pointers
From: Van Snyder
References: 00-245

1 Introduction

Maybe I should stop reading 7.5.2. Every time I do, I find problems with pointer assignment – this time mostly with the constraints. The general problem is that "type" is used without qualifying it to the case of data pointers. The term "type" is also apparently intended to refer to the result type specified in the interface for procedure pointers or procedures, but that's not the same as the type of the pointer – procedure pointers don't have types. The constraints would also be easier to understand, and could be simplified slightly, if they were grouped according to whether they apply to all pointer assignments, only for data, or only for procedures.

2 Edits

Edits refer to 00-007r3. 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 for the editor are noted in the margin, or appear between [and] in the text.

[Editor: Before "an" insert "is a data object that is". This edit makes the constraint more precise; it may not be *necessary*, because a procedure pointer, even one that returns an array, is not an array. Nonetheless, it is helpful.] 139:10

[Editor: Move the constraint at [139:23] to here.] 139:10+

The following constraints apply only if *pointer-object* is not a procedure pointer:

The next constraint is intended to replace the one at [139:32-33], which is defective in not including "procedure pointer" in the conditions for *target*.) *Note to J3 Not an edit*

Constraint: The *target* shall not be a *procedure-name*, a procedure pointer, or an *expr* the result of which is a procedure pointer.

[Editor: Replace "If ... object," by "The".] 139:13

[Editor: Replace "If ... it" by "The *pointer-object*" and insert "the" before "*target*".] 139:15

[Editor: Delete – Replaced in the edit for [139:32-33] below.] 139:19-20

The following constraints apply only if *pointer-object* is a procedure pointer: 139:22

[Editor: After making the changes indicated for [139:28] below, move the constraint at [139:28-31] to here.]

[Editor: "pointer" ⇒ "POINTER" (cf. [139:13])] 139:23

[Editor: Remove "is ... that". Before "the" insert "*target* shall have an explicit interface, and".] 139:24

[Editor: Replace "If ... pointer," by "The". Replace "an" by "a nonelemental".] 139:28

[Editor: The constraint at this place is superceded by one introduced by the edit for [139:10+] above. Replace it by the following:] 139:32-33

Constraint: If *pointer-object* is a subroutine procedure pointer, *target* shall not be a function, a function procedure pointer, or an *expr* the result of which is a function procedure pointer.

Constraint: If *pointer-object* is a function procedure pointer, *target* shall not be a subroutine, a subroutine procedure pointer, or an *expr* the result of which is a subroutine procedure pointer.

It would at first seem that the above two upside-down and inside-out constraints could be simpler, but we have procedure pointers and external entities that are neither known to be subroutines nor known to be functions. The constraints are also not covered by the one at [139:24-27] because that one applies only if the *pointer-object* has explicit interface. *Note to J3 Not an edit*

Constraint: If *pointer-object* is a function procedure pointer and a result type is specified for *target*, the characteristics of the result types (12.2.2), other than the values of nonkind type parameters, shall be the same.

[Editor: After "*pointer-object*" insert "is a data object that" twice.] 139:41,43

[Editor: Replace "If *target* is a disassociated pointer" by:] 139:44

If *pointer-object* is a data object or a function procedure pointer, the following conditions apply. In the case of a function procedure pointer, the term "type" applies to the result type of the function.

- When *target* is a disassociated pointer or an expression the result of which is a disassociated pointer

---

[Editor: Replace "Otherwise" by:] 139:46

- When *target* is an associated pointer or an expression the result of which is an associated pointer

---

[Editor: Put a bullet in front of "If".] 140:3

[Editor: An *expr* doesn't have an association status. After "status" insert "nor shall the result of *expr* have undefined association status".] 140:4

In addition to pointer assignment, a data object with the POINTER attribute becomes associate with a target by execution of an ALLOCATE statement (6.3.1), and becomes disassociated from a target by execution of a NULLIFY statement (6.3.2) or a DEALLOCATE statement (6.3.3). 140:19-20

3 Radical alternative

[Editor: After the end insert "Any previous association between the pointer object and a target is severed."] 139:3

There are two kinds of pointer assignment. Data object pointer assignment affects the pointer association status of a data object that has the POINTER attribute. Procedure pointer assignment affects the pointer association status of a procedure pointer. 139:4-140:4

R735 *pointer-assignment-stmt* **is** *data-pointer-asg-stmt*
or *procedure-pointer-asg-stmt*

7.5.2.1 Data object pointer assignment

R735a *data-pointer-asg-stmt*	**is**	*pointer-object* [(*bounds-spec-list*)] => *target*	
R736 *bounds-spec*	**is**	*lower-bound* :	No change
R737 *target*	**is**	*variable*	
	or	*expr*	

Constraint: The *pointer-object* shall have the POINTER attribute.
Constraint: The *pointer-object* shall not be a procedure pointer.
Constraint: The *variable* shall shall not be a procedure or a procedure pointer, nor shall the result of *expr* be a procedure pointer.
Constraint: If *pointer-object* is not an array a *bounds-spec-list* shall not be specified.
Constraint: If *bounds-spec-list* is specified, the number of *bounds-specs* shall equal the rank of *pointer-object*.
Constraint: The ranks of *pointer-object* and *target* shall be the same.
Constraint: The *target* shall not be an array section with a vector subscript.
Constraint: The *pointer-object* shall be type compatible (5.1.1.8) with the *target*.
Constraint: Corresponding kind type parameters of *pointer-object* and *target* shall have the same values.
Constraint: The *variable* shall have the TARGET attribute or the POINTER attribute.
Constraint: The result of *expr* shall have the POINTER attribute.

If *pointer-object* is not polymorphic (5.1.1.8), *target* shall have the same dynamic type as *pointer-object*. Same as 139:41-43

If *pointer-object* is polymorphic, it assumes the dynamic type of *target*.

The extent of a dimension of *pointer-object* is the extent of the corresponding dimension of *target*. If the lower bound is d and the extent of the corresponding dimension of *target* is s, then the value of the upper bound is $s + d - 1$. If a *bounds-spec-list* is present, it specifies the lower bounds; Otherwise, the lower bound of each dimension is the result of the intrinsic function LBOUND (13.17.58) applied to the corresponding dimension of *target*. Same as 140:12-16

In addition to pointer assignment, a data object with the POINTER attribute becomes associated with a target by execution of an ALLOCATE statement (6.3.1), and becomes disassociated from a target by execution of a NULLIFY (6.3.2) or DEALLOCATE (6.3.3) statement.

7.5.2.2 Procedure pointer assignment

In this subclause, the type and type parameters of a function or of a function procedure pointer are the type and type parameters of the result.

R737a *procedure-pointer-asg-stmt*	**is**	*pointer-object* => *procedure-target*
R737b *procedure-target*	**is**	*target*
	or	*procedure-name*

Constraint: The *pointer-object* shall be a procedure pointer.
Constraint: Corresponding kind type parameters of *pointer-object* and *procedure-target* shall have the same values.
Constraint: The *procedure-name* shall be the specific name of an external, module, or dummy procedure, a specific intrinsic procedure listed in 13.16 and not marked with a bullet (•).
Constraint: The *target* shall be a procedure pointer, an expression the result of which is a procedure pointer, or a reference to the NULL intrinsic function.
Constraint: The *procedure-target* shall not be a nonintrinsic elemental procedure.
Constraint: If *pointer-object* has an explicit interface, *procedure-target* shall have an explicit interface, and the characteristics listed in 12.2 shall be the same for *pointer-object*

and *procedure-target*, except that a target that is pure can be assigned to a *pointer-object* that is not pure, and a target that is an elemental intrinsic procedure may be assigned to a *pointer-object* (which cannot be elemental).

Constraint: If *pointer-object* is a subroutine procedure pointer, *procedure-target* shall not be a function, a function procedure pointer, or an *expr* the result of which is a function procedure pointer.

Constraint: If *pointer-object* is a function procedure pointer, *procedure-target* shall not be a subroutine, a subroutine procedure pointer, or an *expr* the result of which is a subroutine procedure pointer.

It would at first seem that the above two upside-down and inside-out constraints could be simpler, but we have procedure pointers and external entities that are neither known to be subroutines nor known to be functions. The constraints are also not covered by the one at [139:24-27] because that one applies only if the *pointer-object* has explicit interface. *Note to J3 Not an edit*

Constraint: If *pointer-object* is a function procedure pointer and a result type is specified for *procedure-target*, the characteristics of the result types (12.2.2), other than the values of nonkind type parameters, shall be the same.

If the *procedure-name* is the name of a specific intrinsic procedure that is also a generic name, only the specific intrinsic procedure is associated with the *pointer-object*.

7.5.2.3 Requirements common to all pointer assignments

In this subclause, the type and type parameters of a function or of a function procedure pointer are the type and type parameters of the result.

If the *target* or *procedure-target* is not a pointer, the pointer assignment statement associates the *pointer-object* with the *target* or *procedure-target*. If the *target* is a pointer that is associated, or is an *expr* the result of which is a pointer that is associated, the *pointer-object* is associated with the target of that pointer. If the *target* is a pointer that is disassociated or a reference to the NULL intrinsic function, the *pointer-object* becomes disassociated. If the *target* is a pointer with undefined association status, the *pointer-object* acquires an undefined association status. Same as 139:36-40

If *pointer-object* is a data object or a function procedure pointer, the following conditions shall apply:

- When the *target* is a disassociated pointer or an expression the result of which is a disassociated pointer, all nondeferred type parameters of the declared type of *pointer-object* that correspond to nondeferred type parameters of the *target* shall have the same values as the corresponding type parameters of the *target*.
- When the *target* is an associated pointer or an expression the result of which is an associated pointer, all nondeferred type parameters of the declared type of *pointer-object* shall have the same values as the corresponding type parameters of the *target*.
- If *pointer-object* has nondeferred type parameters that correspond to deferred type parameters of the *target* or *procedure-target*, the *target* or *procedure-target* shall not be a pointer with undefined association status, nor shall the result of *expr* have undefined association status.

[Editor: Delete] 140:11-16

[Editor: Delete] 140:19-22