Subject: Elemental operations on pointer association status
From: Van Snyder

1 Number

TBD

2 Title

Elemental operations on pointer association status.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Provide for elemental operations on pointer association status.

6 Rationale

Elemental operations on pointer association status would be useful. The DIS has syntax for it and lacks a constraint against it (hmmm, maybe C614 does constrain against it), but it has no interpretation of that syntax. We already have some forms of elemental operations on pointer association status indirectly by way of intrinsic assignment. Consider

```
type T; real, pointer :: P; end type T
type(t) :: X(100)
real, target :: R
x = t(r)
```

The last statement associates the P component of every element of X with R. One could consider FORALL a more direct but wordy way of doing this:

```
type T; real, pointer :: P; end type T
type(t) :: X(100)
real, target :: R
forall ( i = 1, 100 ) x(i)%p => r
```

One may therefore ask "If the standard already provides it indirectly, why provide it directly?" The answer is that the first indirect form only does exactly what is desired in the case that the type `T` has no other components. If `T` has other components, those components in `X` will also be filled from `Y`; that may not be what is desired. The FORALL method is wordy and requires either implicit or an explicit declaration of an index variable that might have no other purpose (but see 04-155).

It would be clearer simply to be able to write

```
x%p => r
```

7 Estimated Impact

Minor, both for the standard and implementors. Probably at level 4 on the N1594 scale.

8 Detailed Specification

Allow NULLIFY to nullify a pointer component in every element of an array-valued structure. Allow DEALLOCATE to deallocate a pointer (or allocatable?) component in every element of an array-valued structure. Allow pointer assignment to broadcast a pointer association status to a pointer component of every element of an array *variable*.

8.1 Sample edits – to show magnitude of change

---

or *variable* % *pointer-component-name* 114:1-2

C634$\frac{1}{2}$ (R634) The *pointer-component-name* shall be a component of the declared type of *variable* that has the POINTER attribute. 114:3+

If the *variable* is an array, the component named by *pointer-component-name* in every element of that array becomes disassociated. 114:5+

C720 (R735) If *bounds-remapping-list* is specified, *data-target* shall have rank one; otherwise, *data-target* shall have the same rank as *variable-name* or *data-pointer-component-name*. 143:21-22

If *variable* is an array, the effect is as if the pointer assignment were executed separately for each element of the array. 144:8+

If *variable* is an array, the effect is as if the pointer assignment were executed separately for each element of the array. 144:35+

It looks like nothing special is needed for masked array assignment.

A little more work (maybe doubling the amount shown) is necessary to allow deallocating a pointer (or allocatable) component in every element of an array, not least being exceptions to C614 at [105:12-13]. A distinction could perhaps be drawn along the same lines as for subobjects at [105:17-20]. Alternatively, a syntax term other than *allocate-object* could be used, similarly to the scheme proposed here for NULLIFY.

9 History