****

J3/02-295r3

Date: November 2002
To: J3
From: Aleksandar Donev
Subject: Selecting the ''declared'' component of polymorphic objects
Reference: J3-007R3, J3/02-294

****

NOTE: Some small edits that came up while looking at this paper are in J3/02-332.

Summary

Polymorphic objects of CLASS(My_Type) need to occasionally be used in places where a non-polymorphic object of the declared type TYPE(My_Type) is needed. The present standard allows these to be intermixable in certain occasions, but not in others.

The underlying idea in Fortran 2002 is that the perfectly safe type casts up the inheritance tree are done at compile time using the parent component selection mechanism. Type casts down the tree are done with the run-time security of SELECT TYPE. There are gaps in this mechanism, some of which are addressed below.

J3 Response

J3 will more carefully consider the issue of when polymorphic variables need to be explicitly cast into non-polymorphic ones, and suggest how to clarify this in and improve the consistency of the draft standard if needed.

The problem

At present, it is impossible to cast a polymorphic variable into a nonpolymorphic one of its declared type. At the same time, it is possible to cast it into a nonpolymorphic variable of any of the types the declared type extends using the parent component selection mechanism. This is inconsistent with the basic model (substitute) for type casting in Fortran 2002. Neither parent component nor SELECT TYPE provide this.

Furthermore, I believe that one should not be allowed to treat a polymorphic variable as a non-polymorphic variable in any context without an explicit cast. This is at present allowed in argument association and in intrinsic assignment, but not in pointer assignment. Note that the reverse, being able to treat nonpolymorphic variables as polymorphic ones, is essential to polymorphism and should remain as is in the standard.

The inability to treat (cast) a polymorphic variable into a nonpolymorphic one of the declared type is an inconsistency that needs to be addressed. The issue on how this should be done will be further considered by J3. The explict syntax for this should be alike parent component selection, as you suggest, however, issues of association and conflicts with parent component selection need to be considered more carefully.

Proposed Solution

We should provide a way to cast a CLASS(My_Type) variable into a TYPE(My_Type) variable, and that can be used as needed. Introducing a ''declared'' component (better term lacking at present), much alike the parent component, is a natural solution:

```
TYPE, EXTENSIBLE :: My_Type
   ...
END TYPE My_Type
```

```
CLASS(My_Type), ... :: polymorphic_variable

TYPE(My_Type) :: plain_variable
TYPE(My_Type), POINTER :: plain_pointer
INTERFACE
   SUBROUTINE MyProcedure(dummy)
      TYPE(My_Type), INTENT(INOUT) ::  dummy
   END SUBROUTINE
END INTERFACE
```

All of these are now valid:

```
plain_variable=polymorphic_variable%My_Type
plain_pointer=>polymorphic_variable%My_Type
CALL MyProcedure(plain_dummy=polymorphic_variable%My_Type)
```

This simplifies and makes the syntax more consistent.
In my opinion, we should also delete some provisions which do allow the mixing of TYPE(My_Type) and CLASS(My_Type), listed below, but this is still an open issue pending further discussion.

Edits and Choices

These have some flaws in them and so this is only a preliminary suggestion:

Additions:

Every polymorphic variable should have a component named after its declared type, even if it is not of an extended type. So 54:13 should be replaced with (possibly split into two sentences, one at 54:6-):

A scalar object of an extensible type has a scalar, nonpointer, nonallocatable, declared/parent component with the type and type parameters of the declared/parent type. The name of this component is the declared/parent type name.

Something needs to be said about the association of the declared component here, much alike inheritance association for the parent component.

There is the possibility of only allowing such declared component selection for polymorphic variables (see Note 74: 5.7).

Optional changes:

Either,
A)
139:5 add: "variable and expression shall not be polymorphic"
265:9 Delete: "If the dummy argument is allocatable or pointer,"
or
B)
Allow the target in a pointer assignment to be polymorphic, even if the pointer is not polymorphic, i.e., delete C716 on 142:35 (also see C717 on 143:1).