To: J3
From: Malcolm Cohen
Subject: Issues 285 and 286
Date: 11th December 2000

1. Introduction

Issue 285 notes that we need the concept, when establishing an association at runtime, of which is the pre-existing entity the association is being established to, and which is the (possibly new) entity.

The difference being these two is that it is the pre-existing entity which supplies the:
- definition status (and value)
- allocation status (and bounds and value if allocated)
- pointer association status (and bounds if associated).

We'll do this by listing each dynamic association possibility and specifying in each case which entity is which.

Note that there is no need to worry about associations which conceptually last for the entire execution of the program.

Issue 286 says
"Section 14.6.1 lists inheritance association as a form of name association. It seems anomalous ..."

No it doesn't. The word "inheritance" does not appear in 14.6.1. 14.6.0 lists "inheritance association" as a separate form of association - separate from "name", "pointer" and "storage".

The editor is confused. The document is not anomalous here (practically everywhere else, but not here!).

2. Dynamic association happens when ...

(1) Argument association
(2) Host association
(3) Construct association (SELECT TYPE, ASSOCIATE)

Note that we need to cover host association to describe recursive procedures correctly. We cannot just pretend that host associations last for the entire program execution. However, USE association, COMMON association and storage association (EQUIVALENCE) last for the entire execution duration, even when the entities in question come into and go out of existence. (It explicitly says this for USE association at [353:36-37]).

We don't need to worry about pointer associations, because operations on pointers that affect the target already explicitly affect the target. We don't need to worry about inheritance associations, because they are not dynamic (they are defined by the extended type definitions) and so can be considered to last for the entire program execution (whether the objects do or not).

3. Edits to 00-007r3

[106:39-42] Delete paragraph.
{This attempts to handle establishment of an association for allocatable entities. We'll cover this in a new section of chapter 14 which covers all relevant dynamic associations.}

[107:1-12] Delete J3 note 285.

[357:25-26] Replace sentence with
"When a pointer becomes associated with another pointer by argument association, construct association, or host association, the effects on its association status are specified in 14.6.5."
{Replace non-rigorous specification by forward reference to rigorous one. Note removal of inheritance association because inheritance association is not a dynamic association.
NOTE TO EDITOR: 14.6.5 means whatever the new section inserted below becomes (probably 14.6.5).}

[357:29-30] Delete J3 note 286.
{The document is not anomalous here.}

{Embedded comments to J3 in the next edit appear left-justified within {{...}}.}
[360:33+] Add new section
"14.6.5 Establishing Associations

When an association becomes established between two entities, either by argument association, host association, or construct association, certain characteristics of the <<associating entity>> become that of the <<pre-existing entity>>.

For argument association, the associating entity is the dummy argument and the pre-existing entity is the actual argument. For host association, the associating entity is the entity in the host scoping unit and the pre-existing entity is the entity in the contained scoping unit.
{{Comment: The next sentence ought to be unnecessary, but I couldn't find where we specify it at the moment...}}
If the host scoping unit is a recursive procedure, the pre-existing entity that participates in the association is the one from the innermost procedure instance that invoked, directly or indirectly, the contained procedure. For construct association, the associating entity is the associate name and the pre-existing entity is the selector.

When an association is established by argument association, host association, or construct association, the following applies:

- If the associating entity has the pointer attribute, its pointer association status becomes the same as that of the pre-existing entity. If the pre-existing entity has a pointer association status of associated, the associating entity becomes pointer-associated with the same target, and has the same bounds as the pre-existing entity if it is an array.
- If the associating entity has the allocatable attribute, its allocation status becomes the same as that of the pre-existing entity. If the pre-existing entity is allocated, the bounds (if an array), values of deferred type parameters, definition status, and value (if defined) of the associating entity become that of the pre-existing entity. If the associating entity is polymorphic and the pre-existing entity is allocated, its dynamic type becomes the same as that of the pre-existing entity.

{{The bit about dynamic type is not needed for non-allocatables, because the definition of dynamic type (p71) covers them already.}}

If the associating entity is neither a pointer nor allocatable, its definition status and value (if defined) become that of the pre-existing entity. If the entities are arrays and the association is not argument association, the bounds of the associating entity become the same as those of the pre-existing entity."
{Add text to describe what happens when one of these types of association is established. We don't describe what happens with array bounds for argument association (other than for pointers and allocatables) because it is already covered in chapter 12 and is too gruesome for words.}