J3/01-162r1

To: J3
Date: 19th March 2001
Subject: Unresolved issues 5 and 6
From: Malcolm Cohen

1. Introduction

Unresolved issue 6 states:
"We need to specify ... how allocatable components interact with the term "subobject"."

The editor goes on to state that he prefers that the contents of allocatable components should be considered to be subobjects (but only when the component is currently allocated).

It further states:
"Also need to consider the relationship between subobjects and ultimate components."

This is particularly relevant because we need to specify

- how allocatable components interact with the term "subobject"
- how allocatable components interact with the term "ultimate component"
- how the terms "subobject" and "ultimate component" interact so that all three are consistent with each other.

The Technical Report applies to Fortran 95, in which the "subobject" term was ill-defined; in particular, contents of pointers were considered to be subobjects - and therefore so were contents of allocatable components (there was no special edit in this area). In the Technical Report, an allocatable component is like a pointer component in that it is an ultimate component.

Issue 5 just says that we should review ultimate and direct components, and update their glossary entries (and maybe delete direct components altogether).

2. Allocatable components and ultimate components: background.

The current definition of ultimate component is [38:33-34]

> "Ultimately, a derived type is resolved into **ultimate components** that are either of intrinsic type or are allocatable or pointers."

Note that it is the "derived type" that is broken up into ultimate components. Ultimate component-ness is a compile-time concept. Various constraints throughout the document depend on it being a compile-time concept.

If an allocatable component is not an ultimate component, we have problems when it is not allocated. Either we have the situation where:
(1) The "ultimate components" of an object do not sometimes exist. This seems contrary to the concept of an ultimate component, which is intended to capture what a derived-type object is comprised of.
or:
(2) Which components are "ultimate" varies at runtime. This appears to be even more contrary to the concept of an ultimate component.

If we are to keep our concept of ultimate component, it seems that we ought to make allocatable components "ultimate" (just as in the Technical Report).

The example to consider is

```
  TYPE T1
    REAL A
    LOGICAL B
  END TYPE
  TYPE T2
    COMPLEX X
    TYPE(T1),ALLOCATABLE :: Y
    CHARACTER Z
  END TYPE
```

The question you should ask is "what are the ultimate components of T2?".
If ultimate components do not stop at allocatables, there is no answer, because we do not know how many pairs of REAL and LOGICAL components to insert in between the COMPLEX and CHARACTER ones.

3. Allocatable components and subobjects

It seems desirable that the contents of an allocatable component should be considered to be subobjects, so that the definition status of an object with an allocatable component depends on the definition status of that component (unlike the pointer situation). Otherwise it would be possible to modify the allocatable component of an INTENT(IN) variable. Since "subobject" is a term applied to objects (unlike ultimate component which is applied to a type) it is not completely unreasonable for the subobjects to vary from one object to another (of the same type).

4. Another problem with ultimate components

Ultimate components as currently defined do not work at all with parameterised derived types. They could be made to work in a friendly way with kind type parameters, but not with unkind ones. (NB: They do not work even with kind type parameters at the moment.)

The example is similar to the allocatable one, viz:

```
  TYPE PDT(N)
    INTEGER X
    TYPE(T1) Y(N)        ! Type T1 from the first example
    DOUBLE PRECISION Z
  END TYPE
```

with the question "what are the ultimate components of PDT?".

Problems are located in the definition itself (no ultimate decomposition possible if a component is an array of derived type whose size depends on the unkind type parameter). 14.6.2.1.2 and 14.6.2.1.3 (pointer association status) may also be problematic.

It appears that the only way of keeping ultimate components is to say that they stop on components with an array specification that depends on a derived type parameter. Ugh. Note that according to 4.5.2, we would not gain anything from treating kind type parameters in a more friendly manner.

(A careful reading of 4.5.2 says that in

```
   INTEGER,PARAMETER :: WP = KIND(0.0)
   TYPE T1(K)
     SEQUENCE
     REAL(K) X
   END TYPE
   TYPE T2
     SEQUENCE
     REAL(WP) X
   END TYPE
   TYPE(T1(WP)) A
   TYPE(T2) B
```

that A and B are of different type. They both happen to have exactly one component, and this component is of type REAL(WP) named X in both cases, but they nonetheless cannot be mixed.)

Basically, this means that components that depend in an unkind way on derived type parameters are basically like allocatable components, except that the compiler generates the allocation and deallocation requests based on the user's expressions, instead of the user having ALLOCATE and DEALLOCATE statements. In order for the term "ultimate component" to have a well-defined meaning, we may have to make this obvious in the standard.

5. Direct components

The term "direct component" is only used to determine the effects of default initialisation. It seems that default initialisation is meant to work with parameterised derived types in the obvious way. Ugh. (We should make an example of this.) This means that our current definition is totally unworkable. We'll need instead to define default initialisation in terms of subobjects.
After attempting to define default initialisation in terms of subobjects, I decided that I needed two "helper" terms to cut down on the acres of verbiage.
First off, (noun) "subcomponent" - a subobject that is a component.
Secondly, (adjective) "default-initialized" - applicable only to subcomponents, indicates that it should be default initialized. The term "default initialized" does not itself appear in the standard except in a note.

6. Other

While reading the text to work out where to put this stuff, I found various trivial problems. Edits to fix these are included below, marked with *TRIVIA*.

7. Recommendations

(1) That ultimate components continue to stop at allocatables.
(2) That we remove the term "direct component".
(3) That we introduce the terms "subcomponent" and "default-initialized".
(4) That we reword default initialisation so that it works.
(5) That subobjects retain their current definition.
(6) That the ultimate parameterised derived type problem be left to another paper.

8. Edits to 01-007

[38:24-27] Delete the definition of "direct component".

[38:28-32] Delete J3 note 5.

[44:38] Delete
"The <initialization-expr> is evaluated in the scoping unit of the type definition."
{The EXTENDS clause already is in the scoping unit of the type definition. We just want the usual default scoping rules here.}

[44:38-40] Replace "The evaluation ... <initialization-expr>." by
"If necessary, the value is converted according to the rules of intrinsic
assignment (7.5.1.5) to a value that agrees in type, type parameters and shape
with the component."
{Replace strange-looking text with the same incantation we use elsewhere to achieve this effect.}

[44:46+] Add new paragraph
"A subcomponent (6.1.2) is **default-initialized** if the type of the object of which it is a component specifies default initialization for that component. If a subcomponent of an object is default-initialized, no subcomponent of that component is default-initialized (any default initialization of such subcomponents has been overridden by the default initialization of the higher subcomponent)."
{Define our adjective. This might be slightly redundant with the preceding paragraph.}

[45:26] After "MEMBER" insert "(NAME_LEN)".
[45:27] Change "20" to "NAME_LEN", and change "NAME" to ":: NAME = "".
{Make the example - which otherwise mostly duplicates 4.23/4.24 - more interesting. In particular, it now has a pdt with a default-initialized dynamic component. Urk.}

TRIVIA
[53:39-40] Replace with
"A binding that has the NON_OVERRIDABLE attribute in the parent type shall not be overridden.".
{Fix bad grammar (passive/active verb tense confusion) by rephrasing sentence.}

TRIVIA
[54:32+] Add new sentence to end of paragraph:
"If necessary, the value is converted according to the rules of intrinsic assignment (7.5.1.5) to a value that agrees with the type parameters of the type parameter."
{Usual incantation, modified to fit (we already know that it is integer scalar).}

TRIVIA
[54:33-34] Delete.
{Typo "acording", frameism "724" and poor wording (see previous edit).}

[95:19-29] Delete J3 note 6.

[96:7+] New paragraph
"A **subcomponent** of an object of derived type is a component of that object, or a component of a subobject of that object."
{Our new noun.}

[358:1-2] Replace "direct ... provided that the"
by "default-initialized subcomponents of"
{Subcomponents that are initially defined.}

[359:26-27] Replace item text with
"Allocation of an object that has a nonpointer default-initialized subcomponent causes that subcomponent to become defined."
{Default-initialise on ALLOCATE.}

[359:34-36] Replace "and is ... those components" by
"causes all nonpointer default-initialized subcomponents".
{Default-initialise unSAVEd local variables on entry.}

[359:37-39] Replace "derived ... that component" by
"causes all nonpointer default-initialized subcomponents".

{Default-initialise INTENT(OUT) args.}

[359:40-42] Replace ", in which ... that component" by
"causes all nonpointer default-initialized subcomponents".
{Default-initialise function result variables.}

[361:18] Replace "direct ... specified" by
"default-initialized subcomponents of the argument".
{INTENT(OUT) thingy not undefined if default-initialised.}

[361:24-25] Replace "direct ... specified" by
"default-initialized subcomponents of the result".
{Function result not undefined if default-initialised.}

[402:14+] Insert new definition
"**default-initialized** (4.5.1.2): A subcomponent is said to be default-initialized if it will be initialized by default initialization."

[402:34-37] Delete the glossary entry for "direct component".

TRIVIA
[409:6] Replace typo "*co*" by "object designator".
{Fix typo from 00-007r3.}

[409:6+] Insert new definition
"**subcomponent** (6.1.2): A subobject that is a structure component."