To: J3
Subject: Lack of PROCEDURE() considered harmful.
From: Malcolm Cohen.
Date: 2002-08-13

1. Introduction

It's always tempting to add new features at the last minute. Paper 02-166r2 did exactly that at the last meeting. Experience has shown us that almost always things get broken when features are added at the last minute. Unfortunately, 02-166r2 is not an exception to this rule.

My objections to 02-166r2 are in three parts:
(1) It is too complicated. Things that are figured out by "automatic compiler magic" are more complicated than things which are explicitly specified.
(2) It is no longer clear that binding labels (NAME=) work.
(3) It has destroyed its own justification (in C.9.3).
(4) We now appear to have generics where specifics might be abstract interfaces instead of procedures.
(5) It appears to be self-inconsistent, or at least very poorly defined.

Items (2) and (3) would require further edits to the standard to fix.

With item (4), we appear to have nonsense semantics. I find it hard to believe that this is not inadvertent. If it is deliberate, I'd certainly like to hear some explanation; if it is accidental, it needs to be fixed.

This is certainly no guarantee that there is not further breakage caused by this last-minute addition.

Therefore I propose that we essentially revert to the pre-meeting-161 state of the abstract interface facility.

2. Philosophical

In F90 and F95 we knew what interface bodies declared - external (or dummy) procedures. Everything was fine. We even built Fortran implementations which used this information to provide the user with high-quality error detection (used mostly for access to libraries of F77-like code, but nonetheless useful).

Now, we don't know what an interface body is. It might be an external (or dummy) procedure, or it might not. The test for telling which is might be is not trivial: we need to examine its usage throughout the whole scoping unit. Now, the compiler might be doing some of this anyway, to see whether to generate a linker reference, but it's pretty bad for the poor user. Looking at the code, he cannot easily tell whether something is an external procedure or an abstract interface.

And meanwhile, we need to turn off our error checking. What a pain. And people who provide interface blocks for their library of external procedures don't get it checked any more unless they actually write calls to every routine therein.

I really strongly object to this. If we're going to have these things, they should be properly declared with keywords and all, not just happening by "compiler magic". The trouble with compiler magic is that no-one ever understands what is going on. And the change is incompatible with existing implementations (yes, I know it's not incompatible with existing valid programs - by a kluge - but implementations and software tools matter too, you know).

3. The good idea

The plausible idea in paper 02-166r2 was to allow procedures with explicit interfaces to be used as interface definitions for PROCEDURE statements et al. This is good because it is fairly obvious what is happening, and it avoids having to make up abstract interfaces when there are already obvious candidates from which to take the interface information (e.g. module procedures).

But that does not mean that we don't want some transparent way of distinguishing between external procedures and abstract interfaces. I might not have been a great fan of the INTERFACE PROCEDURE() syntax but it sure beats not having anything at all.

4. Self-inconsistency? NAME=?

[257:19-20] says
"An interface body specifies an explicit specific interface."

I note that we don't have explicit interfaces as independent entities, they are interfaces "of a procedure" (see [255:6]).

I can see several possible interpretations here:
(1) an interface body actually specifies an abstract interface; this is what some of the text inserted by 02-166r2 seems to think; and that if a referenced procedure has the same name as that of an abstract interface, it has the same characteristics and dummy argument names.
In this case, NAME= does not work, because the binding label is not part of the abstract interface.

(2) an interface body specifies an interface to a procedure (as much of the pre-existing text in ch12 thinks). In this case, NAME= works, but the new text in ch5 does not seem to make much sense.

(3) if a procedure reference appears (later) in a scoping unit, any procedure body with that name defines a procedure interface (*NOT* an abstract interface - it potentially has extra attributes), and if no such reference appears, it defines an abstract interface.

5. Subclause C.9.3

Subclause C.9.3 currently states
[464:36-37]
"A scoping unit is allowed to contain an interface body for a procedure that does not exist in the program, provided the procedure described is never referenced."

This is complete nonsense with the "magic abstract" feature: an interface body which is never referenced as a procedure is now an abstract interface (a local entity) and not an external procedure declaration. Indeed, the entire rest of this paragraph only makes sense if you can declare an external procedure with an (unreferenced) procedure body.

To make C.9.3 consistent with the magic abstract feature, we would have to delete [464:36-42].

It has been pointed out that although C.9.3 is highly desirable, and consistent with the (previous) rest of the standard, it lacked normative force.

6. Notes

There are many edits in 02-166r2 which have nothing to do with the purported topic of that paper, and for which no justification is provided. Some of these are addressed by other papers (e.g. the EXTERNAL attribute); this paper only addresses the changes made to remove the PROCEDURE() syntax.

In the grand rewrite tradition, some edits improve the descriptions and do not simply reinstate the 02-007r1 text.

The "good idea" (see above section 3) is not altered by this paper, i.e. we propose to keep the current ability to use any procedure with an explicit interface in the PROCEDURE(...) statement et al.

7. Proposed TECHNICAL CHANGEs

1. That the explicit syntax requirement for abstract interfaces be reinstated.

2. That abstract interfaces are introduced by "ABSTRACT INTERFACE". This wording is less ugly and more intuitive than the previous one. (The keyword ABSTRACT here is like a qualifier for INTERFACE, similar to PURE for FUNCTION.)

3. That C.9.3 (see above) be given normative force.

8. Edits to 02-007r2

[256:13+] insert "<<or>> ABSTRACT INTERFACE"
{Reinstate syntax requirement, but with prettier syntax.}

[256:30+] Insert
"C1202a (R1203) If the <interface-stmt> is ABSTRACT INTERFACE, then the <function-name> in the <function-stmt> or the <subroutine-name> in the <subroutine-stmt> shall not be the same as a keyword that specifies an intrinsic type."
{Reinstate constraint.}

[257:18-] Insert at beginning of paragraph:
"An interface block introduced by ABSTRACT INTERFACE is an <<abstract interface block>>. An interface body in an abstract interface block specifies an <<abstract interface>>.
{Reinstate semantics.}

[257:19] Break paragraph after "<<specific interface block>>".

[257:19] Insert new paragraph into break:
"The name of the entity declared by an interface body is the <function-name> in the <function-stmt> or the <subroutine-name> in the <subroutine-stmt> that begins the interface body."
{Simplify wording.}

[257:19-22] Replace "An interface ... by the interface body." with
"An interface body in a generic or specific interface block specifies the EXTERNAL attribute and an explicit specific interface for an external procedure or a dummy procedure. If the name of the declared procedure is that of a dummy argument in the subprogram containing the interface body, the procedure is a dummy procedure; otherwise, it is an external procedure. An interface body in an abstract interface block specifies an abstract interface."
{Reword and reintroduce abstract interfaces.}

[257:23] After "explicit interface" insert "or abstract interface".
{Reinstate abstract interfaces.}

[257:30-31] After "body" insert
"An explicit specific interface may be specified by an interface body for an external procedure that does not exist in the program if the procedure is never used in any way."
{Make the library-friendly part of C.9.3 normative. The "used in any way" is to capture all potential uses which might require the implementation to generate a linkage: this includes normal references, argument association, procedure pointer association and type-bound procedure specification.}

Replace BNF term "abstract-interface-name" with "interface-name" throughout
chapters 4 and 12 (it doesn't occur anywhere else), because it no longer

refers only to abstract interfaces.
{NOTE: This should be done even if the rest of this paper is not.}

[262:13-15] Replace first two sentences with
"C1211 (R1215) The <name> shall be the name of an abstract interface or of a procedure that has an explicit interface."

{Note: C1212 should remain.}

[263:Note 12.14 line 1] Before "INTERFACE" insert "ABSTRACT".
[263:Note 12.14 line 5+] After "END FUNCTION REAL_FUNC" insert
"END INTERFACE
INTERFACE"
{Reinstate abstract interface into example.}

[277:5] Before "a dummy" insert "an abstract interface or".
{Reinstate constraint.}

[394:6] Replace "interface bodies" with "abstract interfaces".

[394:11-12] Delete "or interface body name (12.3.2.1),"

[399:14] Replace "interface bodies" by "abstract interfaces".

[399:34] Replace with
"The name of an entity declared by an interface body".

[417:2] After "procedure," insert "abstract interface,".

[464:37] After "referenced" insert "or used in any other way".
{Expand "referenced" to include other legitimate uses.}

A. Annex: Edits of 02-166r2 which do not impinge on this question

For information only, here is a list of edits in 02-166r2 which do not impinge on the abstract interface syntax question and which are therefore not altered by this paper.

Part 1: Edits which are ok
- [xiii, item (6)]
- [254:13-14]
- [255:5+]
- [255:7-8]
- [263:2-3]

Part 2: Edits which are addressed by other papers
- [80:23]
- [80:23-26]
- [80:36-38]