Date: 9 October 1998
To: J3
From: Van Snyder
Subject: Comments on Richard Maine's comments in 98-204 and 98-205 concerning 98-186r1

This is an attempt to remedy, at least in part, some of the unresolved issues mentioned in 98-204 and 98-205. Unless otherwise specified, references are to 98-007r3.

1 Issues raised in 98-204

1.1 Fifth paragraph concerning edits at 40:22 in 98-007r2

The bnf term *binding-by-name* was originally used when attempting to unify type-bound procedures and type-bound operators. Since the usefulness of type-bound operators for extensible types depends on changes in the rules for generic resolution, and no such changes have yet been made, there were no edits in 98-186r1 to support type-bound operators. The extra level of syntax rules remained after syntax rule alternatives to support type-bound operators were removed.

1.2 Third paragraph concerning edits at 48:28+ in 98-007r2

The sentence "The binding inherited from the parent is not accessible in objects of the type" provided a definition for the term "overrides." After removing it, there is nothing left that expresses the same restriction, except implications derived from the generic meaning of "overrides," or the usage of the term with respect to default initialization, which is incorrect in this instance.

Perhaps "accessible" is the wrong term, but we need to have something explicit.

1.3 Paragraph concerning edits at 225:11 in 98-007r2

In the re-wording of 12.4.1, at 239:34, "theh" should be "the". At 239:47, add "(4.5.1)" after "passed-object dummy argument." Edits

1.4 Paragraph concerning edits at 308:33 in 98-007r2

The proposal to add "subobjects" to the section heading 14.1.2.5 derived from earlier objections, in 98-168, that the "component name" inherited from the parent type isn't a component – it's a subobject of the extended type.

1.5 Paragraph concerning edits at 346:37+ in 98-007r2

This was intended to be another definition for "override." At 365:15, in the same paragraph, add "(4.5.3) If a procedure is bound to an extensible type by the same *binding-name* as one that would be inherited from the parent type, it *overrides* the one that would be inherited from the parent type." Edit

2 Issues raised in 98-205

2.1 Unresolved issue 35 on page 39

At 39:3 replace "CONTAINS" by "*contains-stmt*." Edit

There is no "*private-stmt*." There is "*access-stmt*" but this includes "PUBLIC" and allows a list of items on which the accessibility attribute is conferred. There is "*private-sequence-stmt*," but this includes "SEQUENCE," which it is not intended to allow here. Should R422 be repaired by splitting "*private-sequence-stmt*" into two parts, or by referring to "*access-stmt*" with a constraint that it shall be "PRIVATE" and can't name any entities upon which the PRIVATE attribute is conferred? No matter how it's repaired, it's necessary that the constraint at 39:26-27 continue to apply. Perhaps the constraint should be repeated at 41:31+: Edit?

Constraint: An *access-spec* (5.1.2.2) is permitted only if the type definition is within the specification part of a module.

At the end of the sentence at 252:28, add ", or it separates the declaration of components of a derived type from the declaration of procedure bindings (4.5.1)." Edit

2.2 Unresolved issue 36 on page 41

If the paragraphs about PASS_OBJ at 41:7-9 and 41:32-34 are converted to constraints, is it permitted for one of them to contain the definition of "passed-object dummy argument?" It is possible to move this definition to a new section 4.5.1.x (not 4.5.1.5 because it applies to pointer components, too), but it would be a very small section.

2.3 Unresolved issue 37 on page 41

Instead of the constraint that the POINTER attribute shall be specified (which got left out – see edits for 98-007r2 at 40:22+ in 98-186r1) and a constraint that no attribute may be specified more than once, it is simpler to do it all with syntax rules:

R434 *proc-component-def-stmt* **is** PROCEDURE([*proc-interface*]), ■ Edits
■ *proc-component-attr-spec* :: *proc-decl-list*

R435 *proc-component-attr-spec* **is** [PASS_OBJ,] POINTER
or POINTER, PASS_OBJ

At 40:39+, add: Edit

Constraint: If *proc-interface* consists of *type-spec* it shall specify an intrinsic type or any accessible derived type including the type being defined.

At 41:25, replace *binding-attr* by *binding-attr-list*. Otherwise, it's not possible to have more than one *binding-attr*, and the following constraint isn't needed. Edit

At 41:31+, add: Edit

Constraint: No *binding-attr* shall be specified more than once.

At 72:37, add "accessible" before "abstract" and at 72:37+ add: Edit

Constraint: The *type-spec* shall specify an intrinsic type or any accessible derived type.

2.4 Unresolved issue 40 on page 47

I don't understand the argument "Every type is accessible" in the note. This seems to contradict the first paragraph of 4.5.1.6. Maybe the constraint at 39:28-29 needs to be re-worded as well.

(This constraint appears also at 33:12-13 in 1539:1991.) I agree that the essence of the second sentence of the first paragraph of 4.5.1.5 (at 45:24-26) should be explained in 4.5.1.6. I couldn't find whether we made any statement about accessing public entities of private type not making the type accessible. If not, maybe we don't need the third sentence of this paragraph. If so, then the third sentence is a parallel construction. In any case, remove the second sentence of the paragraph at 45:24-26. Edit

2.5 Unresolved issue 42 on page 52

The intent and effect of the restriction at 51:34-35 is that a non-pure type-bound procedure can be overridden by a pure one, but not vice-versa. This is consistent with argument association and pointer assignment – a pure procedure can be called using an impure interface, but not vice-versa.

2.6 Unresolved issue 43 on page 53

At 47:22, add "`! POINT ACCESSED BY HOST ASSOCIATION.`" Edit

At 53:15, change `POINT3D` to `POINT_3D` and add Edit
"`! POINT_3D ACCESSED BY HOST ASSOCIATION.`"

2.7 Unresolved issue 44 on page 48

At 48:1, replace "If a type definition contains a PRIVATE statement" by "If a PRIVATE statement is specified before CONTAINS in a type definition." Edit

Replace the paragraph at 48:9-11 by "If no PRIVATE statement is specified following CONTAINS, the default accessibility of procedure bindings is PUBLIC. If a PRIVATE statement is specified following CONTAINS, the default accessibility of procedure bindings is PRIVATE. The default accessibility may be superceded by an accessibility attribute of a *proc-binding*. Procedure bindings that are private are accessible only within the module containing the type definition, even if the type itself is public (5.1.2.2)." Something similar to this will presumably be necessary at 48:1 if we get around to allowing mixed private and public data components. (We already use "override" for two similar meanings; I think it's better to use it again, with a similar meaning, than to introduce the term "supercede" – for the same reason that introducing the term "annotation" was undesirable.) Edit

I don't think it's *necessary* from a language design point of view, but it probably simplifies thingss if at 53:4+ we add Edit

(6) Either both shall be private or both shall be public (4.5.1.6).

2.8 Unresolved issue 45 on page 238

There's nothing anywhere in 12.4.0 about procedure invocation. I don't understand what's special in this place about references to type-bound procedures. Do we need to say something in general in 12.4.0 about procedures being invoked?

Replace 238:20 by "The specific procedure denoted by *binding-name* is the one specified by a *proc-binding* in the declaration of the dynamic type of *data-ref*, or inherited (4.5.3.1) into the declaration of the dynamic type of *data-ref* if none is declared. See also **4.5.3.2 Type-bound procedure overriding**. Edit

Concerning 238:21, I would prefer that it were not possible to instantiate objects of types that have deferred bindings, thereby converting the run-time prohibition to a constraint. There's a small loss of functionality, but it's no tragedy – similar in magnitude to the loss of functionality one suffers by eschewing unspecified intent.

2.9 Unresolved issue 46 on page 46

None of the mess of type-bound procedure nomenclature, or type-bound procedure reference would be necessary if dynamic procedure polymorphism were implemented by extension of genericity. There would be other problems instead, but these might be easier to solve. Such a change is probably impracticable at this time, so we probably need to muddle the mess somewhat to make it work. A possibility not mentioned in the J3 note about unresolved issue 46 is a "binding of a procedure to a type."

2.10 Unresolved issue 47 on page 240

At 240:19-21, replace the sentence beginning "In a procedure"' by "In a procedure reference in which *variable* consists of *structure-component* and the final *part-name* is a procedure pointer with the PASS_OBJ attribute, the object of which the *part-name* is a component is associated, as an actual argument, with the passed-object dummy argument." Edit

2.11 Unresolved issue 48 on page 323

Replace 323:1-5 by "A binding name has the same scope as the type to which it is used to bind a procedure. Outside of the type definition, it may appear only as the *binding-name* part of *data-ref%binding-name* within a *call-stmt* or *function-reference*. If the type is accessed in a scoping unit by use association (14.6.1.2) and the accessibility (4.5.1.6) of the binding name is public, or the type is accessed in a scoping unit by host association (14.6.1.3), then the binding name is accessible for use as the *binding-name* part of *data-ref%binding-name* within a *call-stmt* or *function-reference*. Edit

Add a J3 internal note: "Do we want to say that a binding name has the scope of a derived type instead of saying that it has the same scope as the type to which it binds a procedure? Edit

Is the sentence at 322:36-39 precisely correct? It appears to imply that private components are not accessible if the type is accessed by host association. J3 question

2.12 Unresolved issue 49 on page 323

Perhaps the last sentence of the first paragraph suggested to remedy unresolved issue 48 should be "If an object of the type is accessible and the binding name is accessible (4.5.1.6), then the binding name may be used as the *binding-name* part of *data-ref%binding-name* within a *call-stmt* or *function-reference*." Edit

2.13 Unresolved issue 50 on page 360

Is this an improvement? Replace 360:4-5 by "**binding** (4.5.1.5): A name that is declared within a type definition for the purpose of providing access to a specific procedure." Edit

Does this require expanding the definition of "access?" Should a different term be used?

2.14 Unresolved issue 51 on page 365

Is the following an improvement? Replace 365:16-18 by "**passed-object dummy argument** (4.5.1): If a specific procedure is accessible by a binding to a type, and the binding has the PASS_OBJ attribute, then the first dummy argument that has the type in which the binding is declared is the **passed object dummy argument** when the specific procedure is accessed by that binding. A specific procedure may have several passed object dummy arguments if it is bound to different types." Edit

Does this require expanding the definition of "access?" Should a different term be used?

2.15 Unresolved issue 52 on page 368

Is the following an improvement? Replace 368:5-6 by "**type-bound procedure** (4.5.1.5): A procedure that is accessible by using a binding declared within a type definition. The procedure is called *type bound* because access to the procedure by way of a binding with public accessibility (4.5.1.6) cannot be excluded or replaced if the type is accessible." Edit

Does this require expanding the definition of "access?" Should a different term be used?