Subject: Comments, questions and miscellaneous edits concerning 99-007r2
References: 97-218r2, 98-181, 98-221, 99-007r2, 99-106r2, 99-107, 99-136
From: Van Snyder

Change bars indicate differences between 99-182r2 and 99-182r3.

1 Problems, questions, comments

There is no pretense offered that remarks in this section constitute complete edits necessary to correct problems or answer questions noted here. Some of the alleged problems may not be problems at all. Some of them should perhaps turn into unresolved issues. Minor specific edits are offered in the next section.

Page and line numbers refer to 99-007r2.

The preference to use syntax terms instead of descriptive names begs for a way to get their definitions into the index. I don't think it's necessary to index every appearance in every syntax rule – indexing the left-hand-sides would be enough.	everywhere
There is no normative definition of entity.	somewhere
Is an ELEMENTAL procedure OK? I can't think of a reason why not, but it does bear pondering.	44:15-19, 33-37
We probably need a constraint that all of the nonkind parameters of the passed-object dummy argument shall be assumed.	44:26+
The note suggests that a dummy function can have a result with a character length parameter of "*". This is not currently allowed by normative text at 74:16-24, which are specified at 74:15 to be the only ways allowed. In light of the constraint at 74:1-2, it seems the list is incomplete. Also see remarks at 270:1ff.	74:37-38
The rules about the relation of ALLOCATABLE to INTENT should be spelled out here, as are already the rules about the relation of POINTER to INTENT.	77:3-20
The term "data object" excludes functions, function procedure pointers and dummy functions. Does it make sense to inquire about nondeferred parameters of the result types of these entities? If we allow assumed parameters for the result type of a dummy function procedure pointer or dummy function (see remark at 270:1ff below), it would at least be useful to inquire about them. I can't find a definition for the term "inquire about." Is this intended to work by analogy with "inquiry function?"	104:2
I'm curious why "properties" is used here, but "type parameters and bounds" is used in a parallel sentence at 142:1.	139:13
One should be able to learn here that using NULL() for *target* results in nullifying the *pointer-object*, but one can't except indirectly in the case of assignment to a procedure pointer.	7.5.2
I didn't know that expressions "delivered" anything.	155:38
Do we need to say anything special about passed-object dummy arguments?	156:2+
Are any of the problems in unresolved issue 3 (203:10-31) reduced by writing "The input/output operation terminates?" Similarly, are any of those problems reduced by writing "The input transfer terminates" at (223:36, 224:7)?	223:24

The term "code" has no definition. Add it to the glossary, or use a different term.	255:23

Should it be permitted for the result type of a dummy function or dummy function procedure pointer to have an assumed type parameter? Allowing an asterisk for the length parameter of the character result type of a dummy function is printed in small type at 74:1-2. If we go so far as to allow a dummy function or dummy function pointer result type to have a type parameter that is a specification expression (not just an initialization expression), I don't see any extra difficulty in the additional step of allowing the result type to be declared to have an assumed type parameter, so long as it is spelled out that the parameter value is assumed from the associated actual argument's function declaration, not somehow assumed into the function associated as an actual argument from the context of its invocation. If it's OK, we also need a constraint at or near 270:23+: 270:1ff

Constraint: If *proc-interface* is present and declares the procedure entities to be functions, and the result type has an assumed type parameter, the functions shall be dummy procedures or dummy procedure pointers.

I don't object if this changes the semantics of dummy functions that have character type result with assumed length, because it was apparently proposed that this be deleted altogether.

I agree that the result of a non-dummy function or function procedure pointer should not be permitted to have an assumed nonkind parameter, except for the "grandfathered" case of character length. This is different, however, from the case of deferred type parameters of a function result, which indicate that the function sets those parameters – this is not a problem because such a result is required to be allocatable or a pointer.

The same remarks apply to assumed shape of the result of a dummy function. That is, if we allow it, the shape is assumed from the associated actual argument, not somehow assumed into the function when it is invoked.

Do we need to say anything here about pointer or allocatable actual arguments suffering from actions taken on dummy arguments? Or, indeed, from actions on any associated entity, e.g. does the type selector in a SELECT TYPE construct suffer from actions taken on the associate name? Do we need to add "or any associated entity" in lots of places in section 14? 14.6.2.1.2-3

2 Edits

Edits refer to 99-007r2. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + indicates that immediately following text is to be inserted after the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: "compatability" ⇒ "compatibility"]	3:12, 403:9
[Editor: ".." ⇒ "."]	13:25
[Editor: Correct the constraint:] Constraint: If EXTENDS or EXTENSIBLE is present, BIND(C) shall not be present.	41:39-40
[Editor: Correct the constraint at 41:39-40:] Constraint: If EXTENDS or EXTENSIBLE is present, SEQUENCE shall not be present.	42:35+
[Editor: Replace "never" by "not."]	45:19

"Never" could be construed to refer to a temporal sequence, as is the usual interpretation of "when."	*Note to J3*
[Editor: Replace "never" by "not," and delete "always."]	46:32-33
"Never" and "always" could be construed to refer to a temporal sequence, as is the usual interpretation of "when."	*Note to J3*
[Editor: Insert "dynamic" before "type". Insert "or of an unallocated entity" after "disassociated".]	75:36
[Editor: "function" ⇒ "procedure" twice.]	83:2-3
[Editor: Delete "still"]	125:19
[Editor: "When" ⇒ "If".]	153:22
[Editor: Replace ".." by "."]	222:14
[Editor: After "specific interface," add "if the specific interface is accessible"]	266:19
[Editor: "dring" ⇒ "during".]	275:42
[Editor: Replace "it" by "they", and "it is" by "they are" twice – refers to "type and type parameters".]	284:43
[Editor: Add ", allocatable," after "valued"]	284:45
[Editor: Change 6.4 to 14.6.2.1.2–3.]	378:31