Subject: Starting to rebuild finalization
From: Van Snyder

1 Introduction

This paper attempts to start rebuilding finalization. I don't have earlier drafts with me, so some of the syntax might be slightly different from what was originally adopted. This can be easily corrected (or the new syntax adopted if it is now preferred).

2 Edits for finalization

Edits refer to 00-007. 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 (before) the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

or FINAL 44:21+

Constraint: If FINAL is specified, no other *binding-attr* shall be specified and *binding* shall not be NULL(). 44:22+

[Editor: Add at the end of the constraint: "If FINAL is specified the procedure shall be a subroutine that has one dummy argument with a declared type of *type-name* and that is polymorphic if and only if *type-name* is extensible. This argument shall not have INTENT(OUT), nor have the ALLOCATABLE, OPTIONAL, POINTER or VALUE attribute. All nonkind parameters of the dummy argument shall be assumed."] 44:31

Constraint: If several subroutines are bound to the type with *binding-attr* FINAL, they shall be distinguished according to the rules for unambiguous procedure references (14.1.2.3). 44:31+

A type bound subroutine that is specified with a *binding-attr* of FINAL is a **final subroutine** for objects of the type. A final subroutine may be elemental. The final subroutines that are bound to the type or that are inherited (4.5.3.1) from the parent type and not overriden (4.5.3.2) are considered to be specific interfaces of a generic interface with a unique processor-dependent name. 49:14-

If the dynamic type of an object has one or more final subroutines, and one is selected according to the rules for resolving procedure references to names established to be generic (14.1.2.4.1), it is invoked before any object of the type is deallocated (6.3.3, 6.3.3.1) or becomes undefined by the events specified by items (3) or (13)(c) in 14.7.6, with the object as its actual argument. If the subroutine causes objects of the type to be deallocated or to become undefined by the events specified by items (3) or (13)(c) in 14.7.6, it shall be recursive.

Immediately following execution of a final subroutine, if it overrides (4.5.3.2) one, the overridden final subroutine is invoked, with the object as its actual argument. This process is repeated until no further final subroutine is available.

Immediately following this process, the object becomes deallocated or undefined.

($2\frac{1}{2}$) Either both shall have the FINAL *binding-attr* or neither shall. 54:19+

If a final subroutine (4.5.1.6) cannot be distinguished from one inherited (4.5.3.1) from the parent type according to the rules for unambiguous procedure references (14.1.2.3), it overrides that subroutine. 54:45+

3 Needed in any case

[Editor: Replace "PASS_OBJ" by "a PASS_OBJ *binding-attr*" because it's a forward reference.] 44:14

[Editor: Insert "other than a SELECT TYPE (8.1.4.1) or ASSOCIATE (8.1.5.1) statement" after "statement".] 107:21

If a SELECT TYPE (8.1.4.1) or ASSOCIATE (8.1.5.1) statement references a function whose result is allocatable or a structure with a subobject that is allocatable, and the function reference is executed, an allocatable result and any subobjects that are allocated allocatable entities in the result returned by the function are deallocated after execution of the construct. 107:24+

When an intrinsic assignment statement (7.5.1.5) is executed, allocatable components of the *variable* are deallocated before the assignment takes place. 107:27+