Insert new subclause:

**4.9 Polymorphism**

Fortran supports object orientation with single inheritance. A derived type may be extended to form a new type with all the components of its parent type and may have additional components. The extended type also has a parent component with the name of the parent and the type and type parameters of the parent. A variable may be declared as polymorphic; it has a declared type and a dynamic type that may be any extension of the declared type. A type declaration may declare that procedures are bound to the type; each has a binding name that may be the same as its procedure name and usually has a dummy argument of the parent type that is given the pass attribute. A type-bound procedure is invoked as if it were a component of the object; if the procedure has an argument with the pass attribute, the corresponding actual argument is omitted from the argument list and the invoking object is passed automatically. Binding names are inherited by extensions of the type but may be overridden by a specification for the same name in the definition of an extended type. Which procedure is invoked in a type-bound reference is determined by the dynamic type of the object through which the procedure is referenced. To execute alternative code depending on the dynamic type of a polymorphic entity and to gain access to the dynamic parts, the select type construct is provided.

**6.4.1 Applicability to language**

Para 1, line 2. Change “parts” to “much”.

Para 2, line 2. Change “the rounding mode can be changed” to “most processors support the rounding mode being changed”.

Para 2, line 3. After “this rounding mode” add “is required to be supported and”.

Para 3, line 1. Delete “the limits of”.

**6.4.2 Guidance to language users**

Bullet 5. Change “limits” to “properties”.

Bullet 7. Move the clause “where the IEEE intrinsic modules and the IEEE real kinds are in use” to the start of the sentence and change “limits” to “extent”.

Bullet 8. Move the clause “where the IEEE intrinsic modules are in use” to the start of the sentence.

**6.6.2 Avoidance mechanisms for language users**

Final bullet. Change “not propagated” to “processed appropriately”.

**6.34.2 Avoidance mechanisms for language users**

Bullet 3. Change to “Use a processor or a static analysis tool to check all interfaces.”

**6.42.2 Avoidance mechanisms for language users**

Delete the third bullet. [It was based on a misunderstanding of this vulnerability.]

**6.43.2 Avoidance mechanisms for language users**

Replace the second bullet by

* Monitor the depth of recursion and limit it.
* Declare type bound procedures as non\_recursive if they are not intended ever to be called recursively.
* When overriding a type-bound procedure, check that its uses by other procedures bound to the type are not affected.

[ A Fortran program illustrating this is appended to this file.]

**6.44.2 Avoidance mechanisms for language users**

Add bullet

* Avoid using the intrinsic function transfer to perform an unsafe cast.

[I can’t think of anything more that is special to Fortran.]

**6.54.1 Applicability to language**

Para 5. Replace second sentence “This does not …” by “However, the default initialization of a component of a variable of derived type does not affect the save attribute of that variable.”

**6.59 Concurrency – Activation [CGA]**

Insert title **6.59.1 Applicability to language**

On line 1, after “Fortran” add “during program activation”.

Replace para 3 by

“The construct do concurrent gives permission to execute a set of iterations of a loop body in parallel but the mechanisms by which this is achieved are not specified. It is the responsibility of the implementation to indicate that the image has failed if it is unable to execute the construct.”

Delete para 4. [Essence moved to new subclause.]

Add subclause:

**6.59.2 Avoidance mechanisms for language users**

* Use the avoidance mechanisms of ISO/IEC 24772-1:2019 clause 6.59.5.
* At the start of the program insert a sync all statement with an iostat= specifier to ensure that all images are executing.

**6.63.1 Applicability to language**

At the end of para 1 add

“There are several mechanisms (see clause 4.10) for ensuring that the sequencing of the execution of the images leads to the intended results. It is essential to use one or more of these mechanisms to avoid the disruptions discussed in ISO/IEC 24772-1 clause 6.63.”

Delete para 2.

**6.63.2 Avoidance mechanisms for language users**

Replace bullet 2 by

* Use the mechanisms listed in bullet 3 of Subclause 6.61.2.

**module m**

 **type c**

 **integer depth**

 **contains**

 **procedure :: a,b**

 **end type**

 **type, extends (c) :: cd**

 **contains**

 **procedure :: b => newb**

 **end type**

**contains**

 **subroutine a(arg)**

**! non\_recursive subroutine a(arg)**

 **class (c) arg**

 **arg%depth = arg%depth+1**

 **write(\*,\*) "In a, with depth= ",arg%depth**

 **if (arg%depth>5) stop**

 **call arg%b()**

 **end subroutine**

 **subroutine b(arg)**

 **class (c) arg**

 **write(\*,\*) "In b, with depth= ",arg%depth**

 **end subroutine**

 **subroutine newb(arg)**

**! non\_recursive subroutine newb(arg)**

 **class (cd) arg**

 **write(\*,\*) "In newb, with depth= ",arg%depth**

**! call arg%c%a()**

 **call arg%a()**

 **end subroutine**

**end module**

**program test**

 **use m**

 **class (cd), allocatable :: arg**

 **allocate(arg)**

 **arg%depth = 0**

 **call arg%a()**

**end program test**