Date: 22 July 1998
To: J3
From: Van Snyder
Subject: Uniformitization and simplification of generic declarations
References: 98-179

1 Discussion

Paper 98-179 advocated allowing

- `PROCEDURE ::` *binding-name* `=>` *binding-list*,
- `OPERATOR (` *defined-operator* `) =>` *binding-list*, and
- `ASSIGNMENT(=) =>` *binding-list*

within an extensible type to declare generic type-bound procedures, instead of requiring an interface block. This is admirably terse, yet clear. Is there any reason this could not be allowed outside of derived type definitions, as an alternative to or abbreviation for interface blocks? That is, allow

```
PROCEDURE :: G => S1, S2, S3
```

as a synonym for

```
INTERFACE G
  PROCEDURE :: S1, S2, S3
END INTERFACE G
```

and similarly for `OPERATOR ( .FOO. ) => S4.  S5` and `ASSIGNMENT(=) => S6, S7, S8`

If not, then for consistency, the verbose `INTERFACE ... END INTERFACE` form should be required within type definitions.

It may in any case be desirable to allow the `INTERFACE ... END INTERFACE` form within a type declaration, above the `CONTAINS` statement, to develop a generic object-bound procedure pointer.

2 Related remarks

There seems to be no technical difficulty to extend generic interfaces to include internal procedures, statement functions, intrinsic procedures, and other generic interfaces – subject to a constraint against circular inclusion of one generic within another.

3 Tentative edits to implement suggestions of section 1

Edits refer to 98-007r2. 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.

---

or *generic-interface-decl* [10:8+]

[Editor: After edits at this same place in paper 98-179.] [39:29+]

or *generic-proc-subobject-def*

[Editor: After edits at this same place in paper 98-179.] [40:22+]

R432x *generic-proc-subobject-def* **is** INTERFACE *generic-subobject-name*
proc-component-def-stmt ...
END INTERFACE

Constraint: Each *proc-component-def-stmt* shall specify an explicit abstract procedure interface.

A generic procedure subobject definition establishes a generic interface (12.3.2.1.1) to all procedure pointers declared within the interface block. The interfaces of the procedure pointers shall be distinguishable according to the rules in 14.1.2.3.

If several generic procedure subobject definitions within the same type definition have the same generic subobject name, the effect is as though all procedure subobject definitions contained within them appeared within a single generic procedure subobject definition.

The generic procedure subobject name and the specific procedure pointer component names are accessible as structure components.

The generic procedure subobject name is neither a data object nor a pointer object.

[This edit applies to edits proposed in 98-179, not to 98-007r2. Add "generic interface declarations" to the list twice.] [47:39-44]

[Editor: Add the following within the same sentence.] [48:17]

, except that a generic procedure subobject declaration in an extended type may have the same name as a generic procedure subobject declaration inherited from the parent type, in which case it extends that generic interface.

[Editor: new section.] [222:37]

12.3.2.1.5 Compact syntax for generic procedure declaration

A generic name, defined operation or defined assignment may be declared by using a more compact syntax.

R1207a *generic-interface-decl* **is** [MODULE] PROCEDURE :: *generic-name* ■
■ => *procedure-name-list*
or OPERATOR (*defined-operator*) => ■
■ *procedure-name-list*
or ASSIGNMENT (=) => *procedure-name-list*

Constraint: A *procedure-name* shall have an explicit interface and shall refer to an accessible procedure pointer, external procedure, dummy procedure, or module procedure.

Constraint: If MODULE appears in a *generic-interface-decl*, each *procedure-name* in that statement shall be accessible in the current scope as a module procedure.

A *generic-interface-decl* has the same effect, and the same constraints, as a generic interface block having the same generic name, defined operator symbol, or equals symbol, containing procedure statements that specify the same procedure names.