Subject: Derived-type input/output and its relation to type-bound procedures
From: Van Snyder
References: 00-179

1 Introduction

The predecessor of this paper, 00-179, was presented at meeting 153. The /jor subgroup suggested a slightly different approach from the one presented in that paper, and invited the present paper.

There are several problems with derived-type input/output that could be addressed by a minor extension to the type-bound procedure mechanism: It is possible to access a type from a module, but not access its input/output procedures, and it is not possible to inherit or defer a derived-type input/output procedure.

2 Specifications and syntax

In addition to the interface-block-based mechanism to specify derived-type input/output procedures described in 9.5.4.4.3, allow a variation on the type-bound procedure declaration mechanism to specify a derived-type input/output procedure. E.g.

```
GENERIC :: READ(FORMATTED) => my_read_routine_formatted
```

serves the same purpose as the interface block at [190:26-41]. Obviously similar extensions provide for unformatted input and for output.

This binding cannot be excluded during use association, is inherited into extension types, can be overridden in them, and an obvious variation allows deferred derived-type input/output procedures to be specified.

It is prohibited to have derived-type input/output procedures specified both by type binding and by an interface block for a single type and set of kind type parameters. It is necessary, but not currently specified, that the derived type argument of a user-defined derived-type procedure shall be polymorphic if and only if the type is extensible.

3 Edits

Edits refer to 00-007r2. 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 (before) the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: "*proc-binding*" ⇒ "*proc-binding-stmt*" twice.] 43:20,21

---

or GENERIC [(*proc-interface-name*)] ■ 43:22+
■ [, NON_OVERRIDABLE] :: ■
■ *dtio-generic-spec* => *binding-list*

[Editor: "The" ⇒ "If PROCEDURE is specified, the".] 43:23

Constraint: If GENERIC is specified, the *proc-interface-name* shall be specified if and only if the binding is to NULL().	43:24+
Constraint: If *dtio-generic-spec* is specified, the interface of each binding shall be as specified in 9.5.4.4.3. The type specified for the `dtv` argument shall be the type being defined.	43:28
Constraint: If several bindings are specified for a single *dtio-generic-spec*, their interfaces shall differ as specified in 14.1.2.3.	43:45+

4.5.1.5$\frac{1}{2}$ User-defined derived-type input/output bindings 48:30+

A binding with a *dtio-generic-spec* specifies a user-defined derived-type input/output binding. The use of a binding for a particular *dtio-generic-spec* is described in 9.5.4.4.3. The set of user-defined derived-type input/output bindings for a particular type and *dtio-generic-spec*, including those that are explicitly specified and those that are inherited (4.5.3.1) into that type and not overridden (4.5.3.2), defines a type-bound generic interface, in exact analogy to generic procedure names.

NOTE 4.31$\frac{1}{2}$

A binding may be to a procedure, or may be deferred. If a deferred binding is selected for use during data transfer (14.1.2.4.2$\frac{3}{4}$), an error condition occurs.

[Editor: Before ", and" insert ", type-bound generic interfaces" twice.]	52:40, 42
[Editor: Before "binding" insert "nongeneric".]	53:36
If a generic binding declared in a type definition has the same *dtio-generic-spec* and the same kind type parameters for the derived-type argument as one inherited from the parent type then the binding declared in the type overrides the one inherited from the parent type. Otherwise it extends the type-bound generic interface for the type being defined and the specified *dtio-binding-spec*.	53:38+ new ¶ after note 4.44
[Editor: Before "the" insert "the specified *proc-interface-name* or".]	115:42-43
[Editor: Replace "whose ... section" by "accessible by a *dtio-generic-spec* (4.5.1.5$\frac{1}{2}$, 12.3.2.1)".]	189:26-27
[Editor: Replace "transfered." by "transferred, as described in 14.1.2.4.2$\frac{3}{4}$. If the selection results in a disassociated procedure pointer, a deferred binding, or a dummy procedure that is not present, an error condition occurs."]	189:30
[Editor: Replace "If an interface ... scoping unit" by "If a derived-type input/output procedure is selected as specified in the previous paragraph".]	189:31-32
The procedures are specified to be used for derived-type input/output by interface blocks (12.3.2.1) or by derived-type procedure bindings (4.5.1.5$\frac{1}{2}$), with a *dtio-generic-spec*.	190:24+ no new ¶
The four interfaces for user-defined procedures for input/output of nonextensible types are:	190:25
If the *generic-spec* is `READ ( FORMATTED )` the interface shall be:	190:26
If the *generic-spec* is `READ ( UNFORMATTED )` the interface shall be:	190:41-42
If the *generic-spec* is `WRITE ( FORMATTED )` the interface shall be:	191:2-3
If the *generic-spec* is `WRITE ( UNFORMATTED )` the interface shall be:	191:18-19
[Editor: Delete.]	191:30

For extensible types, the only difference in the interfaces is that the `dtv` argument shall be polymorphic, i.e. declared with a CLASS keyword instead of a TYPE keyword. 191:32+

---

	or *dtio-generic-spec*	246:30
R1207a *dtio-generic-spec*	**is** READ(FORMATTED)	

Constraint: If *generic-spec* is a *dtio-generic-spec*, each specific interface shall differ from the specific interfaces of generic bindings for that *dtio-generic-spec* and the type of the `dtv` argument (9.5.4.4.2) of that interface, as specified in 14.1.2.3. 246:47+

[Editor: Add in the same paragraph:] 251:13+

As with generic names, several subroutines may be defined in interface blocks having a given *dtio-generic-spec*, in which case the *dtio-generic-spec* is generic in exact analogy to a generic procedure name (12.3.2.1).

[Editor: Before ", or" insert ", have the same *dtio-generic-spec*" twice.] 343:12, 18

[Editor: Add a new section after – not a subsection of – 14.1.2.4.2.] 346:22+

14.1.2.4.2$\frac{3}{4}$ Resolving derived-type input/output procedure references

J3 note

Unresolved issue xxx This subclause should be merged into 14.1.2.4.1 when that subclause is repaired to account for defined operators and assignment.

If there is an accessible generic interface (12.3.2.1.3) for the type of data transfer as specified in 9.5.4.4.3, and the derived-type dummy argument of one of its specific interfaces is type-compatible with and has the same kind type parameters as the effective list item (9.5.2), the corresponding procedure or procedure pointer is selected.

If there is a type-bound generic interface (4.5.1.5$\frac{1}{2}$) for the declared type of the effective item and a *dtio-generic-spec* for the type of data transfer, and the kind type parameters of the derived-type argument of one of the accessible bindings thereto have the same values as for the effective item, then the corresponding binding from the type-bound generic interface for the same *dtio-generic-spec* and the dynamic type of the effective item is selected.

NOTE 14.9$\frac{1}{2}$

If the binding is deferred, an error condition occurs.

Otherwise, no procedure is selected for derived-type input/output.