Subject: Resolve generic without invoking a procedure or evaluating arguments
From: Van Snyder
Reference: 04-273

Number

TBD

Title

Resolve generic without invoking a procedure or evaluating arguments

Submitted By

J3

Status

For consideration.

Basic Functionality

Given examplars of actual arguments, resolve a generic name to a specific procedure without invoking the procedure or evaluating its arguments.

Rationale

With care and diligence, one can develop a program so that related sets of variables, constants and function results are parameterized by a single kind type parameter. In order to change the kind of that set of entities, one need only change one named constant's definition — almost: Generic procedures cannot be actual arguments or procedure pointer targets. Thus, if one needs to change the program, in addition to changing the single named constant definition, one needs to find all places where a specific procedure that operates on the entities in question is an actual argument or procedure pointer target, and manually edit those appearances.

It would be helpful to have a facility to resolve a generic name to a specific procedure without evaluating any arguments or invoking a procedure.

Estimated Impact

Minor. Processors already know how to do generic resolution. Estimated at meeting 169 to be at 4 on the JKR scale.

Detailed Specification

Given examplars of actual arguments, resolve a generic name to a specific procedure without invoking the procedure or evaluating its arguments.

There are at least two ways to do this. One is to provide a syntax that is suggestive of procedure reference, but does resolution instead. One possibility for this is to enclose an actual argument list in square brackets or curly brackets instead of round brackets. E.g.,

```
call solver ( myVec, myJacobian, myModel[myVec,myJacobian] )
```

Another is to provide an entity that looks like an intrinsic function but that has the important distinction that its arguments aren't evaluated. Indeed, this entity that has the appearance of a function reference isn't even invoked during program execution. It is entirely resolved to a procedure by the processor during translation. E.g.,

```
  call solver ( myVec, myJacobian, resolve(myModel(myVec,myJacobian)) )
```

Since RESOLVE wouldn't really be a function, it may be desirable to use a different kind of brackets, e.g.,

```
  call solver ( myVec, myJacobian, resolve[myModel(myVec,myJacobian)] )
```

If possible, a provision should be made to resolve a defined operation or defined assignment, e.g., `resolve[a * b]` or `resolve[a = b]`.

It should be possible to resolve a type-bound generic reference, e.g., `resolve[a%b]`, but not if the data entity (`a` in this illustration) is polymorphic.

No matter what syntax is used, it should be allowed to use the result either as an actual argument or a procedure pointer target.

It is conceivable that a provision could be made to resolve a generic name from the context of its appearance. This could work if it is an actual argument associated with a dummy procedure provided that both the referenced procedure and the dummy procedure have explicit interface, or if it is a target in a procedure pointer assignment and the pointer has explicit interface. This would still require some means to cause resolution in the implicit interface cases, so it may not be worth contemplating.

History