Subject: Provide a way to embed decisions within expressions
From: Van Snyder
Reference: 03-258r1, section 2.8.1

## Number

TBD

## Title

Provide a way to embed decisions within expressions.

## Submitted By

J3

## Status

For consideration.

## Basic Functionality

Provide a way to embed decisions within expressions. For example, a distfix IF-THEN-ELSE operator would be useful.

## Rationale

One sometimes needs to select one thing or another to be used within an expression. At present, one creates a temporary variable, sets that variable with an if-then-else or where-elsewhere construct, then evaluates the expression using that variable.

Another use is to compute whether an actual argument is present. This cannot be done by creating a temporary variable. Instead, one uses an if-then-else construct with the argument textually present in one branch but not the other. If one wants to compute whether $n$ actual arguments are present, one needs a complicated if-then-elseif-else construct with $2^n$ branches.

Other languages include a distfix if-then-else operator. For example, in C one can write `p ? x : y`, which is pronounced *if p then x else y*. If such an expression were to be the *target* in a pointer assignment, its result should be a target, not a value. This spelling could work in Fortran as well. Clunky alternatives might be `.IF. p .THEN. x .ELSE. y .ENDIF.`, or more briefly `p .THEN. x .ELSE. y`. For the case of computing whether an actual argument is present, the syntax might be `p ? x`, pronounced *if p then x is the actual argument, else the actual argument is not present.* `p` shall be scalar in this case.

The difference for these operators as compared to existing operators is that only the first operand (`p` in the example) is initially evaluated. Then the second operand (`x` in the example) is evaluated if (where in the elemental case) `p` is true, else (elsewhere in the elemental case) the third operand (`y` in the example) is evaluated if it appears. The first operand is required to be logical, while the others are required to be of the same type, type parameters and rank. If `p` is an array, `x` and `y` have to be the same shape as `p`. In the `p ? x` case, `p` would necessarily have to be a scalar. The result type, type parameters and rank are those of `x`. If `p` is a scalar, the shape of the result is `x` or `y` depending on whether `p` is true or fales. If `p` is an array, the shape of the result is the shape of `p`.

It would introduce substantial complication into the defined-operator discussion to allow to overload this operator. Fortunately, it seems unlikely that would be useful.

The functionality almost exists in the MERGE intrinsic function. The reason it isn't quite the same as what is described here is that the standard specifies that all arguments of a function are evaluated

before the function is invoked. Also, we don't have a two-argument MERGE that causes its result not to exist (for purposes of argument association) if its first argument is false.

## Estimated Impact

Minor.

## Detailed Specification

Provide a way to embed a decision witin an expression, for example, a distfix IF-THEN-ELSE operator.

Providea way to compute whether an actual argument is present.

## History