Mantis 1523

P1800-2012 *sub-clause 11.4.11:*

Motivation

This Mantis clarifies how the semantics for the conditional operator when the *cond_predicate* evaluates to an ambiguous value and the first and second expression is a non-integral type.

Change

If *cond_predicate* is true, the operator returns the value of the first *expression* without evaluating the second expression; if false, it returns the value of the second *expression* without evaluating the first expression. If *cond_predicate* evaluates to an ambiguous value (`x` or `z`), then both the first *expression* and the second *expression* shall be evaluated, and their results shall be combined bit by bit using Table 11-20 to calculate the final result unless either the first or second expression is real, in which case the result shall be `0`. The first and second expressions are extended to the same width, as described in 11.6.1 and 11.8.2.

To:

If *cond_predicate* is true, the operator returns the value of the first *expression* without evaluating the second expression; if false, it returns the value of the second *expression* without evaluating the first expression. If *cond_predicate* evaluates to an ambiguous value (`x` or `z`), then both the first *expression* and the second *expression* shall be evaluated and compared for logical equivalence as described in 11.4.5. If that comparison is true(1), the operator shall return either the first or second *expression*. Otherwise the operator returns a result based on the data types of the expressions.

When both the first and second *expressions* are of integral types, if the *cond_predicate* evaluates to an ambiguous value and the *expressions* are not logically equivalent, their results shall be combined bit by bit using Table 11-20 to calculate the final result ~~unless either the first or second expression is real, in which case the result shall be 0~~. The first and second expressions are extended to the same width, as described in 11.6.1 and 11.8.2.

Change:

"The conditional operator can be used with nonintegral types (see 6.11.1) and aggregate expressions (see 11.2.2) using the following rules:

— If both the first *expression* and second *expression* are of integral types, the operation proceeds as defined.

— If the first *expression* or second *expression* is an integral type and the opposing expression can be implicitly cast to an integral type, the cast is made and proceeds as defined."

to

"The conditional operator can be used with nonintegral types (see 6.11.1) and aggregate expressions (see 11.2.2) using the following rules:

— If both the first *expression* and second *expression* are of integral types, the operation proceeds as defined.

— If both *expressions* are **`real`**, then the resulting type is **`real`**. If one *expression* is **`real`** and the other *expression* is **`shortreal`** or integral, the other *expression* is cast to **`real`**, and the resulting type is **`real`**.

If one *expression* is **`shortreal`** and the other *expression* is integral, the integral *expression* is cast to **`shortreal`**, and the resulting type is **`shortreal`**.

— Otherwise, if ~~If~~ the first *expression* or second *expression* is of an integral type and the other expression can be implicitly cast to an integral type, the cast is made and proceeds as defined above for integral types.

Change

For nonintegral and aggregate expressions, if *cond_predicate* evaluates to an ambiguous value, then:

— If the first *expression* and the second *expression* are of a class data type and if the conditional operation is legal, then the resulting type is determined as defined above and the result is null.

— Otherwise, both the first *expression* and second *expression* shall be evaluated, and their results shall be combined element by element. If the elements match, the element is returned. If they do not match, then the default-uninitialized value for that element's type shall be returned.

To:

For nonintegral and aggregate expressions, if *cond_predicate* evaluates to an ambiguous value and the *expressions* are not logically equivalent, then:

~~— If the first *expression* and the second *expression* are of a class data type and if the conditional operation is legal, then the resulting type is determined as defined above and the result is null.~~

~~— Otherwise, both the first *expression* and second *expression* shall be evaluated, and~~ For aggregate array data types, except associative arrays, where both expressions contain the same number of elements their results shall be combined element by element. If the elements match, the element ~~is~~ shall be returned. If they do not match, then the ~~default-uninitialized~~ value specified in Table 7-1 for that element's type shall be returned.

— For all other data types, the value specified in Table 7-1 for the resulting type as defined above shall be returned.