The Role of Mathematical Equivalence in Fortran

David Muxworthy d.muxworthy@bcs.org.uk 27 September 2012

Arithmetic expressions (F66)

Fortran 66 (6.4):

The evaluation may proceed according to any valid formation sequence (except as modified in the following paragraph).

When two elements are combined by an operator, the order of evaluation of the elements is optional. If mathematical use of operators is associative, commutative, or both, full use of these facts may be made to revise orders of combination, provided only that integrity of parenthesized expressions is not violated. … The associative and commutative laws do not apply in the evaluation of integer terms containing division, hence the evaluation of such terms must effectively proceed from left to right.

A part of an expression need be evaluated only if such action is necessary to establish the value of the expression.

Arithmetic expressions (F77)

Fortran 77 (6.6.4):

Once the interpretation has been established in accordance with those rules, the processor may evaluate any mathematically equivalent expression, provided that the integrity of parentheses is not violated.

Two arithmetic expressions are mathematically equivalent if, for all possible values of their primaries, their mathematical values are equal. However, mathematically equivalent arithmetic expressions may produce different computational results.

Arithmetic expressions -1 (F90)

Fortran 90 (7.1.7.3):

The rules given in 7.2.1 specify the interpretation of a numeric intrinsic operation. Once the interpretation has been established in accordance with those rules, the processor may evaluate any mathematically equivalent expression, provided that the integrity of parentheses is not violated.

Two expressions of a numeric type are mathematically equivalent if, for all possible values of their primaries, their mathematical values are equal. However, mathematically equivalent expressions of numeric type may produce different computational results. For example, any difference between the values of the expressions (1./3.)*3. and 1. is a computational difference, not a mathematical difference.

Arithmetic expressions -2 (F90)

Fortran 90 (7.1.7.3):

Expression	Allowable Alternative Form
X + Y	Y + X
X * Y	Y * X
-X + Y	Y - X
X + Y + Z	X + (Y + Z)
X - Y + Z	X - (Y - Z)
X * A / Z	X * (A / Z)
X * Y - X * Z	X * (Y - Z)
A / B / C	A / (B * C)
A / 5.0	0.2 * A

The same table continues in each standard to F08 (7.1.5.2.4 Note 7.19)

Arithmetic expressions -3 (F90)

Fortran 90 (7.1.7.3):

Expression	Nonallowable Alternative Form
I / 2	0.5 * I
X * I / J	X * (I / J)
I / J / A	I / (J * A)
(X + Y) + Z	X + (Y + Z)
(X * Y) - (X * Z)	X * (Y - Z)
X * (Y – Z)	X * Y - X * Z

In addition to the parentheses required to establish the desired interpretation, parentheses may be included to restrict the alternative forms that may be used by the processor in the actual evaluation of the expression. This is useful for controlling the magnitude and accuracy of intermediate values developed during the evaluation of an expression. For example, in the expression
A + (B - C)
the parenthesized expression (B - C) must be evaluated and then added to A.

The same table and text continues in each standard to F08 (7.1.5.2.4 Notes 7.19 and 7.20)

Relational Expressions (F66)

Fortran 66 (6.2):

A relational expression consists of two arithmetic expressions separated by a relational operator and will have the value true or false as the relation is true or false, respectively.

One arithmetic expression may be of type real or double precision and the other of type real or double precision, or both arithmetic expressions may be of type integer. If a real expression and a double precision expression appear in a relational expression, the effect is the same as a similar relational expression. This similar expression contains a double precision zero as the right hand arithmetic expression and the difference of the two original expressions (in their original order) as the left. The relational operator is unchanged.

Relational Expressions (F77)

Fortran 77 (6.3.3):

If the two arithmetic expressions are of different types, the value of the relational expression

$$e_1 \ \underline{\text{relop}} \ e_2$$

is the value of the expression

$$((e_1) - (e_2)) \ \underline{\text{relop}} \ 0$$

where 0 (zero) is of the same type as the expression $((e_1) - (e_2))$, and relop is the same relational operator in both expressions. Note that the comparison of a double precision value and a complex value is not permitted.

Relational Expressions (F90)

Fortran 90 (7.1.7.5):

The rules given in 7.2.3 specify the interpretation of relational intrinsic operations. Once the interpretation of an expression has been established in accordance with those rules, the processor may evaluate any other expression that is relationally equivalent, provided that the integrity of parentheses in any expression is not violated. For example, the processor may choose to evaluate the expression

```
I .GT. J
```

where I and J are integer variables, as

```
J - I .LT. 0
```

Two relational intrinsic operations are relationally equivalent if their logical values are equal for all possible values of their primaries.

Relational Expressions (F95)

Fortran 95 (7.1.7.5):

NOTE 7.28

For example, the processor may choose to evaluate the expression

```
I .GT. J
```

where I and J are integer variables, as

```
J - I .LT. 0
```

Two relational intrinsic operations are relationally equivalent if their logical values are equal for all possible values of their primaries.

Relational Expressions (F03)

Fortran 03 (7.1.8.5):

NOTE 7.22

For example, the processor may choose to evaluate the expression

NOTE 7.22 (cont.)

```
   I > J
```

where I and J are integer variables, as

```
   J - I < 0
```

Two relational intrinsic operations are relationally equivalent if their logical values are equal for all possible values of their primaries.

Relational Expressions (F08)

Fortran 08 (7.1.5.5.2):

Once the interpretation of a relational intrinsic operation is established, the processor may evaluate any other expression that is relationally equivalent, provided that the integrity of parentheses in any expression is not violated.

Two relational intrinsic operations are relationally equivalent if their logical values are equal for all possible values of their primaries.

That is, Note 7.22 has been deleted

Interpretation F03/0065 - Question

```
Given
  REAL X
  X = ... some value ...
may
  IF( X+3.0 .EQ. 3.0 )...
be transformed into
  IF( X .EQ. 0.0 )...
by the processor?
```

In Fortran 2003, 7.1.8.5 Evaluation of relational intrinsic operations says "Two relational intrinsic operations are relationally equivalent if their logical values are equal for all possible values of their primaries."

On a machine where addition of 3.0 to a small value is not exact, the logical values for X+3.0==3.0 are not the same as X==0.0 for all possible values of X, therefore it would seem that this transformation would not be possible.

However, Note 7.22 in Fortran 2003 shows this transformation as being acceptable.

Interpretation F03/0065 - Answer

ANSWER:

No, the transformation is not permitted unless it gives the same answer for all possible values (of X and X+3.0).

The erroneous example has been removed in Fortran 2008.

EDITS to 10-007r1:

None.

Interpretation F03/0065 - Comments

1. As this parsing takes place at the syntactic level before any computation has started and for continuity with earlier standards, not to mention 'the spirit of Fortran', 'value' should be taken to refer to mathematical values, not computational values.

2. The wording in dates from Fortran 77, but has never truly corresponded with what compilers do. … The effects of this change are not limited to simple examples like this one. For example, many processors have multiple representations of the same value, and it affects which one may be returned.

The question of exactly what a value is needs addressing before an interpretation is appropriate.

3. The general notion of mathematical equivalence was introduced in FORTRAN 77. It has been a bane of Fortran programmers ever since.

4. Are two values different if (a) they are numerically different, (b) they are computationally different or (c) they are representationally different? All three are different concepts in C99 and IEEE 754.

What is a 'value'?

Can this question be resolved in the next revision of the standard?

What should the UK position be?