Subject: Reorganization of Subclause 7.1
From: Van Snyder

Every time I try to read about operations, it's really tedious because they are defined in 7.1.2 and 7.1.3, the interpretations are given in 7.2, and the rules of evaluation are given in 7.1.8. The problem is that the matrix was sliced in the wrong direction. In what follows, I've moved **7.3 Precedence of operators** to be after **7.1.1 Form of an expression**, **7.1.8.1 Evaluation of operations** to be between that and **7.1.2 Intrinsic operations**, each "interpretation" subclause from **7.2 Interpretation of operations** to be after the appropriate definition, and each "evaluation" subclause from **7.1.8 Evaluation** to be after the appropriate "interpretation" subclause. The only things I removed were cross references that became unnecessary. I changed "interpretation of an expression has been established" to "interpretation of the ... operation is established," and the wording of some cross references. If this reorganization is done, it should be done after 06-296 is considered. Here's the new outline for Clause 7:

A clause with only two subclauses is kind of pathetic. Maybe we should split clause **7 Expressions and assignment** into clauses **7 Expressions** and **8 Assignment**, which would increase the clause number of each succeeding clause.

7 Expressions and assignment

[Text is unchanged.]

7.1 Expressions

[Text is unchanged.]

7.1.1 Form of an expression

[Text is unchanged.]

7.1.2 Precedence of operators

[Was **7.3**. Text is unchanged.]

7.1.3 Evaluation of operations

[Was **7.1.8.1**. Text is unchanged.]

7.1.4 Intrinsic operations

An **intrinsic operation** is either an intrinsic unary operation or an intrinsic binary operation. An **intrinsic unary operation** is an operation of the form *intrinsic-operator* x_2 where x_2 is of an intrinsic type (4.4) listed in Table 7.1 for the unary intrinsic operator.

An **intrinsic binary operation** is an operation of the form x_1 *intrinsic-operator* x_2 where x_1 and x_2 are of the intrinsic types (4.4) listed in Table 7.1 for the binary intrinsic operator and are in shape conformance (7.1.9).

The interpretations defined in subclause 7.1.4 apply to both scalars and arrays; the interpretation for arrays is obtained by applying the interpretation for scalars element by element.

The type, type parameters and interpretation of an expression that consists of an intrinsic operation are independent of the type and type parameters of the context or any larger expression in which it appears.

NOTE 7.1

For example, if X is of type real, J is of type integer, and INT is the real-to-integer intrinsic conversion function, the expression INT (X + J) is an integer expression and X + J is a real expression.

Table 7.1: **Type of operands and results for intrinsic operators**

Intrinsic operator *op*	Type of x_1	Type of x_2	Type of $[x_1]$ *op* x_2
Unary +, −		I, R, Z	I, R, Z
	I	I, R, Z	I, R, Z
Binary +, −, *, /, **	R	I, R, Z	R, R, Z
	Z	I, R, Z	Z, Z, Z

Type of operands and results for intrinsic operators (cont.)

Intrinsic operator *op*	Type of x_1	Type of x_2	Type of $[x_1]$ *op* x_2
//	C	C	C
	B	B	B
.EQ., .NE., ==, /=	I	I, R, Z, B	L, L, L, L
	R	I, R, Z, B	L, L, L, L
	Z	I, R, Z, B	L, L, L, L
	C	C	L
.GT., .GE., .LT., .LE. >, >=, <, <=	I	I, R	L, L
	R	I, R	L, L
	C	C	L
.NOT.		L, B	L, B
.AND., .OR., .EQV., .NEQV.	L	L	L
	B	B,I	B
	I	B	B

Note: The symbols I, R, Z, C, L, and B stand for the types integer, real, complex, character, logical, and bits, respectively. Where more than one type for x_2 is given, the type of the result of the operation is given in the same relative position in the next column. For the intrinsic operators with operands of type character, the kind type parameters of the operands shall be the same.

7.1.4.1 Numeric intrinsic operations

A **numeric intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is a numeric operator (+, −, *, /, or **). A **numeric intrinsic operator** is the operator in a numeric intrinsic operation.

7.1.4.1.1 Interpretation of numeric intrinsic operations

The two operands of numeric intrinsic binary operations may be of different numeric types or different kind type parameters. Except for a value raised to an integer power, if the operands have different types or kind type parameters, the effect is as if each operand that differs in type or kind type parameter from those of the result is converted to the type and kind type parameter of the result before the operation is performed. When a value of type real or complex is raised to an integer power, the integer operand need not be converted.

A numeric operation is used to express a numeric computation. Evaluation of a numeric operation produces a numeric value. The permitted data types for operands of the numeric intrinsic operations are specified in 7.1.4.

The numeric operators and their interpretation in an expression are given in Table 7.2, where x_1 denotes the operand to the left of the operator and x_2 denotes the operand to the right of the operator.

Table 7.2: **Interpretation of the numeric intrinsic operators**

Operator	Representing	Use of operator	Interpretation
**	Exponentiation	x_1 ** x_2	Raise x_1 to the power x_2
/	Division	x_1 / x_2	Divide x_1 by x_2
*	Multiplication	x_1 * x_2	Multiply x_1 by x_2
-	Subtraction	x_1 - x_2	Subtract x_2 from x_1
-	Negation	- x_2	Negate x_2
+	Addition	x_1 + x_2	Add x_1 and x_2
+	Identity	+ x_2	Same as x_2

The interpretation of a division operation depends on the types of the operands (7.1.4.1.2).

If x_1 and x_2 are of type integer and x_2 has a negative value, the interpretation of x_1 ** x_2 is the same as the interpretation of $1/(x_1$ ** ABS $(x_2))$, which is subject to the rules of integer division (7.1.4.1.2).

NOTE 7.2

For example, 2 ** (–3) has the value of 1/(2 ** 3), which is zero.

7.1.4.1.2 Integer division

One operand of type integer may be divided by another operand of type integer. Although the mathematical quotient of two integers is not necessarily an integer, Table 7.1 specifies that an expression involving the division operator with two operands of type integer is interpreted as an expression of type integer. The result of such an operation is the integer closest to the mathematical quotient and between zero and the mathematical quotient inclusively.

NOTE 7.3

For example, the expression (–8) / 3 has the value (–2).

7.1.4.1.3 Complex exponentiation

In the case of a complex value raised to a complex power, the value of the operation x_1 ** x_2 is the principal value of $x_1^{x_2}$.

7.1.4.1.4 Evaluation of numeric intrinsic operations

Once the interpretation of a numeric intrinsic operation is established, 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.

NOTE 7.4

Any difference between the values of the expressions (1./3.)*3. and 1. is a computational difference, not a mathematical difference. The difference between the values of the expressions 5/2 and 5./2. is a mathematical difference, not a computational difference.

The mathematical definition of integer division is given in 7.1.4.1.2.

NOTE 7.5

The following are examples of expressions with allowable alternative forms that may be used by the processor in the evaluation of those expressions. A, B, and C represent arbitrary real or complex operands; I and J represent arbitrary integer operands; and X, Y, and Z represent arbitrary operands of numeric type.

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)

NOTE 7.5 (cont.)

X * Y - X * Z	X * (Y - Z)
A / B / C	A / (B * C)
A / 5.0	0.2 * A

The following are examples of expressions with forbidden alternative forms that shall not be used by a processor in the evaluation of those expressions.

Expression	Forbidden 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

The execution of any numeric operation whose result is not defined by the arithmetic used by the processor is prohibited. Raising a negative-valued primary of type real to a real power is prohibited.

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.

NOTE 7.6

For example, in the expression

```
A + (B - C)
```

the parenthesized expression (B – C) shall be evaluated and then added to A.

The inclusion of parentheses may change the mathematical value of an expression. For example, the two expressions

```
A * I / J
A * (I / J)
```

may have different mathematical values if I and J are of type integer.

Each operand in a numeric intrinsic operation has a type that may depend on the order of evaluation used by the processor.

NOTE 7.7

For example, in the evaluation of the expression

```
Z + R + I
```

where Z, R, and I represent data objects of complex, real, and integer type, respectively, the type of the operand that is added to I may be either complex or real, depending on which pair of operands (Z and R, R and I, or Z and I) is added first.

7.1.4.2 Character intrinsic operation

The **character intrinsic operation** is the intrinsic operation for which the *intrinsic-operator* is (//) and both operands are of type character. The operands shall have the same kind type parameter. The **character intrinsic operator** is the operator in a character intrinsic operation.

7.1.4.2.1 Interpretation of the character intrinsic operation

The character intrinsic operator // is used to concatenate two operands of type character with the same kind type parameter. Evaluation of the character intrinsic operation produces a result of type character.

The interpretation of the character intrinsic operator // when used to form an expression is given in Table 7.4, where x_1 denotes the operand to the left of the operator and x_2 denotes the operand to the right of the operator.

Table 7.4: **Interpretation of the character intrinsic operator //**

Operator	Representing	Use of operator	Interpretation
//	Concatenation	x_1 // x_2	Concatenate x_1 with x_2

The result of the character intrinsic operation // is a character string whose value is the value of x_1 concatenated on the right with the value of x_2 and whose length is the sum of the lengths of x_1 and x_2. Parentheses used to specify the order of evaluation have no effect on the value of a character expression.

NOTE 7.8

For example, the value of ('AB' // 'CDE') // 'F' is the string 'ABCDEF'. Also, the value of 'AB' // ('CDE' // 'F') is the string 'ABCDEF'.

7.1.4.2.2 Evaluation of the character intrinsic operation

A processor is only required to evaluate as much of the character intrinsic operation as is required by the context in which the expression appears.

NOTE 7.9

For example, the statements

```
CHARACTER (LEN = 2) C1, C2, C3, CF
C1 = C2 // CF (C3)
```

do not require the function CF to be evaluated, because only the value of C2 is needed to determine the value of C1 because C1 and C2 both have a length of 2.

7.1.4.3 Logical intrinsic operations

A **logical intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is .AND., .OR., .XOR., .NOT., .EQV., or .NEQV. and both operands are of type logical. A **logical intrinsic operator** is the operator in a logical intrinsic operation.

7.1.4.3.1 Interpretation of logical intrinsic operations

A logical operation is used to express a logical computation. Evaluation of a logical operation produces a result of type logical. The permitted types for operands of the logical intrinsic operations are specified in 7.1.4.

The logical operators and their interpretation when used to form an expression are given in Table 7.5, where x_1 denotes the operand to the left of the operator and x_2 denotes the operand to the right of the operator.

Table 7.5: **Interpretation of the logical intrinsic operators**

Operator	Representing	Use of operator	Interpretation
.NOT.	Logical negation	.NOT. x_2	True if x_2 is false
.AND.	Logical conjunction	x_1 .AND. x_2	True if x_1 and x_2 are both true
.OR.	Logical inclusive disjunction	x_1 .OR. x_2	True if x_1 and/or x_2 is true
.EQV.	Logical equivalence	x_1 .EQV. x_2	True if both x_1 and x_2 are true or both are false
.NEQV.	Logical nonequivalence	x_1 .NEQV. x_2	True if either x_1 or x_2 is true, but not both
.XOR.	Logical nonequivalence	x_1 .XOR. x_2	True if either x_1 or x_2 is true, but not both

The values of the logical intrinsic operations are shown in Table 7.6.

Table 7.6: **The values of operations involving logical intrinsic operators**

x_1	x_2	.NOT. x_2	x_1 .AND. x_2	x_1 .OR. x_2	x_1 .EQV. x_2	x_1 .NEQV. x_2	x_1 .XOR. x_2
true	true	false	true	true	true	false	false
true	false	true	false	true	false	true	true
false	true	false	false	true	false	true	true
false	false	true	false	false	true	false	false

7.1.4.3.2 Evaluation of logical intrinsic operations

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

NOTE 7.10

For example, for the variables L1, L2, and L3 of type logical, the processor may choose to evaluate the expression

```
L1 .AND. L2 .AND. L3
```

as

```
L1 .AND. (L2 .AND. L3)
```

Two expressions of type logical are logically equivalent if their values are equal for all possible values of their primaries.

7.1.4.4 Bits intrinsic operations

A **bits intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is //, .AND., .OR., .XOR., .NOT., .EQV., or .NEQV. and at least one operand is of type bits. A **bits intrinsic operator** is the operator in a bits intrinsic operation.

7.1.4.4.1 Interpretation of bits intrinsic operations

For bits intrinsic operations other than concatenation (//), the two operands may be of different types or different kind type parameters. The effect is as if each operand that differs in type or kind type parameter from those of the result is converted to the type and kind type parameter of the result before the operation is performed.

Bit operations are used to express bitwise operations on sequences of bits, or to concatenate such sequences. Evaluation of a bits operation produces a result of type bits. The permitted types of operands of the bits intrinsic operations are specified in 7.1.4.

The bits operators and their interpretation when used to form an expression are given in Table 7.7, where x_1 denotes the operand of type bits to the left of the operator and x_2 denotes the operand of type bits to the right of the operator.

Table 7.7: **Interpretation of the bits intrinsic operators**

Operator	Representing	Use of operator	Interpretation
(//)	Concatenation	x_1 // x_2	Concatenation of x_1 and x_2
.NOT.	Bitwise NOT	.NOT. x_2	Bitwise NOT of x_2
.AND.	Bitwise AND	x_1 .AND. x_2	Bitwise AND of x_1 and x_2
.OR.	Bitwise inclusive OR	x_1 .OR. x_2	Bitwise OR of x_1 and x_2
.EQV.	Bitwise equivalence	x_1 .EQV. x_2	Bitwise equivalence of x_1 and x_2
.NEQV.	Bitwise nonequivalence	x_1 .NEQV. x_2	Bitwise nonequivalence of x_1 and x_2
.XOR.	Bitwise exclusive OR	x_1 .XOR. x_2	Bitwise exclusive OR of x_1 and x_2

The leftmost KIND(x_1) bits of the result of the bits concatenation operation are the value of x_1 and the rightmost KIND(x_2) bits of the result are the value of x_2.

For a bits intrinsic operation other than //, the result value is computed separately for each pair of bits at corresponding positions in each operand. The value of each bit operation, for bits denoted b_1 and b_2 are given in Table 7.8.

Table 7.8: **The values of bits intrinsic operations other than //**

x_1	x_2	.NOT. x_2	x_1 .AND. x_2	x_1 .OR. x_2	x_1 .EQV. x_2	x_1 .NEQV. x_2	x_1 .XOR. x_2
1	1	0	1	1	1	0	0
1	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	0	1	0	0	1	0	0

7.1.4.4.2 Evaluation of bits intrinsic operations

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

NOTE 7.11

For example, for the variables B1, B2, and B3 of type bits, the processor may choose to evaluate the expression

```
B1 .XOR. B2 .XOR. B3
```

as

NOTE 7.11 (cont.)

```
B1 .XOR. (B2 .XOR. B3)
```

Two expressions of type bits are computationally equivalent if their values are equal for all possible values of their primaries.

7.1.4.5 Relational intrinsic operations

A **relational intrinsic operator** is an *intrinsic-operator* that is .EQ., .NE., .GT., .GE., .LT., .LE., ==, /=, >, >=, <, or <=. The operators <, <=, >, >=, ==, and /= always have the same interpretations as the operators .LT., .LE., .GT., .GE., .EQ., and .NE., respectively. A **relational intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is a relational intrinsic operator. A **numeric relational intrinsic operation** is a relational intrinsic operation for which both operands are of numeric type. A **character relational intrinsic operation** is a relational intrinsic operation for which both operands are of type character. The kind type parameters of the operands of a character relational intrinsic operation shall be the same. A **bits relational intrinsic operation** is a relational intrinsic operation for which at least one of the operands is of type bits.

If both operands of a bits relational operation do not have the same kind type parameter, the operand with the smaller kind type parameter is converted to the same kind as the other operand. If one operand of a bits relational operation is not of type bits, it is converted to type bits with the same kind type parameter as the other operand. Any conversion takes place before the operation is evaluated.

7.1.4.5.1 Interpretation of relational intrinsic operations

A relational intrinsic operation is used to compare values of two operands using the relational intrinsic operators .LT., .LE., .GT., .GE., .EQ., .NE., <, <=, >, >=, ==, and /=. The permitted types for operands of the relational intrinsic operators are specified in 7.1.4.

NOTE 7.12

As shown in Table 7.1, a relational intrinsic operator cannot be used to compare the value of an expression of a numeric type with one of type character or logical. Also, two operands of type logical cannot be compared, a complex operand may be compared with another numeric operand only when the operator is .EQ., .NE., ==, or /=, and two character operands cannot be compared unless they have the same kind type parameter value.

Evaluation of a relational intrinsic operation produces a result of type default logical.

The interpretation of the relational intrinsic operators is given in Table 7.9, where x_1 denotes the operand to the left of the operator and x_2 denotes the operand to the right of the operator.

Table 7.9: **Interpretation of the relational intrinsic operators**

Operator	Representing	Use of operator	Interpretation
.LT.	Less than	x_1 .LT. x_2	x_1 less than x_2
$<$	Less than	$x_1 < x_2$	x_1 less than x_2
.LE.	Less than or equal to	x_1 .LE. x_2	x_1 less than or equal to x_2
$<=$	Less than or equal to	$x_1 <= x_2$	x_1 less than or equal to x_2
.GT.	Greater than	x_1 .GT. x_2	x_1 greater than x_2
$>$	Greater than	$x_1 > x_2$	x_1 greater than x_2
.GE.	Greater than or equal to	x_1 .GE. x_2	x_1 greater than or equal to x_2
$>=$	Greater than or equal to	$x_1 >= x_2$	x_1 greater than or equal to x_2
.EQ.	Equal to	x_1 .EQ. x_2	x_1 equal to x_2
$==$	Equal to	$x_1 == x_2$	x_1 equal to x_2

Interpretation of the relational intrinsic operators (cont.)

Operator	Representing	Use of operator	Interpretation
.NE.	Not equal to	x_1 .NE. x_2	x_1 not equal to x_2
/=	Not equal to	x_1 /= x_2	x_1 not equal to x_2

A numeric relational intrinsic operation is interpreted as having the logical value true if and only if the values of the operands satisfy the relation specified by the operator.

In the numeric relational operation

x_1 *rel-op* x_2

if the types or kind type parameters of x_1 and x_2 differ, their values are converted to the type and kind type parameter of the expression $x_1 + x_2$ before evaluation.

A character relational intrinsic operation is interpreted as having the logical value true if and only if the values of the operands satisfy the relation specified by the operator.

For a character relational intrinsic operation, the operands are compared one character at a time in order, beginning with the first character of each character operand. If the operands are of unequal length, the shorter operand is treated as if it were extended on the right with blanks to the length of the longer operand. If both x_1 and x_2 are of zero length, x_1 is equal to x_2; if every character of x_1 is the same as the character in the corresponding position in x_2, x_1 is equal to x_2. Otherwise, at the first position where the character operands differ, the character operand x_1 is considered to be less than x_2 if the character value of x_1 at this position precedes the value of x_2 in the collating sequence (4.4.5.4); x_1 is greater than x_2 if the character value of x_1 at this position follows the value of x_2 in the collating sequence.

NOTE 7.13

The collating sequence depends partially on the processor; however, the result of the use of the operators .EQ., .NE., ==, and /= does not depend on the collating sequence.

For nondefault character types, the blank padding character is processor dependent.

A bits relational intrinsic operation is interpreted as having the logical value true if and only if the values of the operands satisfy the relation specified by the operator.

For a bits relational intrinsic operation, x_1 and x_2 are equal if and only if each corresponding bit has the same value. If x_1 and x_2 are not equal, and the leftmost unequal corresponding bit of x_1 is 1 and x_2 is 0 then x_1 is greater than x_2; otherwise x_1 is less than x_2.

7.1.4.5.2 Evaluation of relational intrinsic operations

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.

NOTE 7.14

For example, the processor may choose to evaluate the expression

```
I > J
```

where I and J are integer variables, as

NOTE 7.14 (cont.)

```
J - I < 0
```

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

7.1.5 Defined operations

A **defined operation** is either a defined unary operation or a defined binary operation. A **defined unary operation** is an operation that has the form *defined-unary-op* x_2 or *intrinsic-operator* x_2 and that is defined by a function and a generic interface (4.5.2, 12.4.3.3).

A function defines the unary operation *op* x_2 if

(1) the function is specified with a FUNCTION (12.6.2.1) or ENTRY (12.6.2.5) statement that specifies one dummy argument d_2,

(2) either

  (a) a generic interface (12.4.3.2) provides the function with a *generic-spec* of OPERATOR (*op*), or

  (b) there is a generic binding (4.5.2) in the declared type of x_2 with a *generic-spec* of OPERATOR (*op*) and there is a corresponding binding to the function in the dynamic type of x_2,

(3) the type of d_2 is compatible with the dynamic type of x_2,

(4) the type parameters, if any, of d_2 match the corresponding type parameters of x_2, and

(5) either

  (a) the rank of x_2 matches that of d_2 or

  (b) the function is elemental and there is no other function that defines the operation.

If d_2 is an array, the shape of x_2 shall match the shape of d_2.

A **defined binary operation** is an operation that has the form x_1 *defined-binary-op* x_2 or x_1 *intrinsic-operator* x_2 and that is defined by a function and a generic interface.

A function defines the binary operation x_1 *op* x_2 if

(1) the function is specified with a FUNCTION (12.6.2.1) or ENTRY (12.6.2.5) statement that specifies two dummy arguments, d_1 and d_2,

(2) either

  (a) a generic interface (12.4.3.2) provides the function with a *generic-spec* of OPERATOR (*op*), or

  (b) there is a generic binding (4.5.2) in the declared type of x_1 or x_2 with a *generic-spec* of OPERATOR (*op*) and there is a corresponding binding to the function in the dynamic type of x_1 or x_2, respectively,

(3) the types of d_1 and d_2 are compatible with the dynamic types of x_1 and x_2, respectively,

(4) the type parameters, if any, of d_1 and d_2 match the corresponding type parameters of x_1 and x_2, respectively, and

(5) either

  (a) the ranks of x_1 and x_2 match those of d_1 and d_2 or

  (b) the function is elemental, x_1 and x_2 are conformable, and there is no other function that defines the operation.

If d_1 or d_2 is an array, the shapes of x_1 and x_2 shall match the shapes of d_1 and d_2, respectively.

NOTE 7.15

An intrinsic operator may be used as the operator in a defined operation. In such a case, the generic properties of the operator are extended.

An **extension operation** is a defined operation in which the operator is of the form *defined-unary-op* or *defined-binary-op*. Such an operator is called an **extension operator**. The operator used in an extension operation may be such that a generic interface for the operator may specify more than one function.

A **defined elemental operation** is a defined operation for which the function is elemental (12.8).

7.1.5.1 Interpretation of a defined operation

The interpretation of a defined operation is provided by the function that defines the operation. The type, type parameters and interpretation of an expression that consists of a defined operation are independent of the type and type parameters of the context or any larger expression in which it appears.

7.1.5.2 Evaluation of a defined operation

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

Two expressions of derived type are equivalent if their values are equal for all possible values of their primaries.

7.1.6 Evaluation of operands

[Was **7.1.8.2**. Text is unchanged.]

7.1.7 Integrity of parentheses

The rules for evaluation specified in subclause 7.1.4 state certain conditions under which a processor may evaluate an expression that is different from the one specified by applying the rules given in 7.1.1 and rules for interpretation specified in subclause 7.1.4. However, any expression in parentheses shall be treated as a data entity.

NOTE 7.16

For example, in evaluating the expression A + (B – C) where A, B, and C are of numeric types, the difference of B and C shall be evaluated before the addition operation is performed; the processor shall not evaluate the mathematically equivalent expression (A + B) – C.

7.1.8 Type, type parameters, and shape of an expression

[Was **7.1.4**. Text is unchanged.]

7.1.9 Conformability rules for elemental operations

[Was **7.1.5**. Text is unchanged].

7.1.10 Specification expression

[Was **7.1.6**. Text is unchanged].

7.1.11 Initialization expression

[Was **7.1.7**. Text is unchanged].

7.2 Assignment

[Was **7.4**. Text is unchanged.]