On Handling of IEEE 754 Exceptions in Interval Relational Operators and Intrinsics

Dmitri Chiriaev and G. William Walster

Draft revised May 17, 1997

Abstract

This note proposes correct nonstop handling of IEEE 754 exceptions in interval relational operators and intrinsics defined in the proposal for interval arithmetic in Fortran 90 [2].

1 Introduction

The issue of properly handling IEEE 754 [1] exceptions in interval arithmetic was first addressed by Popova in [5]. The proposal for interval arithmetic in Fortran 90 [2] defines new interval relational operators and requires that all Fortran intrinsic functions that accept real data shall also accept interval data. Priest [4] defines a set of representable real intervals based on the IEEE floating point number system and offers an implementation of efficient interval arithmetic algorithms that deliver correct results even when exceptions occur. In this note algorithms are defined that provide correct nonstop handling of IEEE 754 exceptions for all the interval intrinsics defined in [2].

2 Valid intervals

Priest's paper[4] defines valid floating point intervals and the real intervals they represent. Since the result of any operation with an empty operand should be empty, Priest proposes to use the interval $[\mathrm{NaN}, \mathrm{NaN}]$ to represent the empty set because if either operand is $[\mathrm{NaN}, \mathrm{NaN}]$, the result will be $[\mathrm{NaN}, \mathrm{NaN}]$.

In addition to $[\mathrm{NaN}, \mathrm{NaN}]$ two additional representations of an empty set are needed: intervals $[x, \mathrm{NaN}]$ and $[\mathrm{NaN}, x]$.

Adding these two represention to the table of valid floating point intervals from Priest's paper[4] we get table 1.

Representation	Interval
$[-0, +0]$	$\{0\}$
$[x, y],\ x \leq y$	$\{z:\ x \leq z \leq y\}$
$[x, -0],\ x < 0$	$\{z:\ x \leq z < 0\}$
$[x, +0],\ x < 0$	$\{z:\ x \leq z \leq 0\}$
$[-0, y],\ 0 < y$	$\{z:\ 0 \leq z \leq y\}$
$[+0, y],\ 0 < y$	$\{z:\ 0 < z \leq y\}$
$[x, +\infty]$	$\{z:\ x \leq z\}$
$[-\infty, y]$	$\{z:\ z \leq y\}$
$[-\infty, -0]$	$\{z:\ z < 0\}$
$[-\infty, +0]$	$\{z:\ z \leq 0\}$
$[-0, +\infty]$	$\{z:\ 0 \leq z\}$
$[+0, +\infty]$	$\{z:\ 0 < z\}$
$[-\infty, +\infty]$	$\mathbf{R}$
$[\mathrm{NaN}, \mathrm{NaN}]$	$\emptyset$
$[x, \mathrm{NaN}],\ x \in \mathbf{R}$	$\emptyset$
$[\mathrm{NaN}, x],\ x \in \mathbf{R}$	$\emptyset$

Table 1: Valid floating point itervals: x and y – any finite, nonzero floating point numbers.

Intervals $[-\infty, -\infty]$ and $[+\infty, +\infty]$, are not allowed to avoid the $\infty - \infty$ invalid operation exception in interval addition.

As in [4] the efficiency of proposed algorithms depends on the efficient implementation of the following properties of the min and max functions:

(i) $\min(x, y) = \min(y, x)$ for all x, y, and similarly for max,

(ii) $\min(x, \mathrm{NaN}) = \max(x, \mathrm{NaN}) = \mathrm{NaN}$ for any x, and

(iii) $\min(-0, +0) = -0$, $\max(-0, +0) = +0$.

See [4] for complete discussion.

3 Interval infix operators

If either operand of the interval intersection or convex hull operators is an empty interval then the result is an empty interval.

```
Z = X.IS.Y          Z <-- intersection of X and Y, that is,
                          [max{xl,yl},min{xu,yu}] if
                          max{xl,yl} < = min{xu,yu} and
                          empty interval [NaN,NaN] otherwise.

Z = X.CH.Y          Z <-- [min{xl,yl},  max{xu,yu}]
```

If an empty interval is an operand of the following operators then FALSE is delivered as the result. The algorithms for these operators proposed in [2] remain unchanged:

```
  X.SB.Y            .TRUE. if X is a subset of Y
                     ( i.e. if xl >= yl .AND. xu <= yu )

  X.PSB.y           .TRUE. if X is a proper subset of Y
                    ( i.e. if X.SB.Y .AND. (xl > yl .OR. xu < yu )

  X.SP.Y            .TRUE. if and only if Y.SB.X is true
                    ( i.e. if xl <= yl .AND. xu >= yu )

  X.PSP.Y           .TRUE. if and only if Y.PSB.X is true
                    ( i.e. if Y.SB.X .AND. (yl > xl .OR. yu < xu )


  R.IN.X            .TRUE. if the REAL value R is contained in the
                    interval X  (i.e. if xl <= R and R <= xu)
```

An empty interval is always disjoint with any other interval. To provide the desired result we must check for `(xl <= xu and yl <= yu)` which is FALSE if one of the intervals is empty.

```
   X.DJ.Y    .TRUE. if X and Y are disjoint sets or one or both of them is empty
         (i.e. if (xl > yu or xu < yl) or  .NOT. (xl <= xu and yl <= yu)
```

Note. It is necessary to use the compliment of the $(xl <= xu$ and $yl <= yu)$ and not to simplify because $((xl > xu)$ or $(yl > yu))$ may not produce the desired TRUE result if a NaN occurs.

4 Interval versions of relational operators

CERTAINLY TRUE relationals

If an empty interval is an operand of a CERTAINLY TRUE relational operator then the result is FALSE. The one exception is the .CNE. operator which returns TRUE in that case.

For example

> ([1,2] .CLT. [3, NaN]) == .FALSE.
>
> [NaN, 3] .CLT. [4,5] == .FALSE.
>
> [1,NaN] .CNE. [1,NaN] == .TRUE.

The algorithms for CERTAINLY TRUE relational operators proposed in [2] are

```
X.CLT.Y       .TRUE. if xu < yl

X.CGT.Y       .TRUE. if xl > yu

X.CLE.Y       .TRUE. if xu <= yl

X.CGE.Y       .TRUE. if xl >= yu
```

To provide the desired result if either argument interval is empty we essentially must add a check for `xl <= xu and yl <= yu` which is FALSE if one of the intervals is empty.

```
X.CLT.Y       .TRUE. if xu < yl  and xl <= xu and yl <=yu

X.CGT.Y       .TRUE. if xl > yu  and xl <= xu and yl <=yu

X.CLE.Y       .TRUE. if xu <= yl and xl <= xu and yl <=yu

X.CGE.Y       .TRUE. if xl >= yu and xl <= xu and yl <=yu
```

However the check can be simplified in this case:

```
X.CLT.Y       .TRUE. if xu < yl  and xl < yu

X.CGT.Y       .TRUE. if xl > yu  and xu > yl

X.CLE.Y       .TRUE. if xu <= yl and xl <= yu

X.CGE.Y       .TRUE. if xl >= yu and  xu >= yl
```

Additional checks should not incure a penalty in the overall performance because relational operations occur relativly seldom.

The .CNE. operator is equivalent to the disjoint operator and as in that case one must check for `(xl <= xu and yl <=yu)` to provide the desired result which is FALSE if one of the intervals is empty.

```
X.CEQ.Y   .TRUE. if  xu <= yl and  xl >=yu,

X.CNE.Y   .TRUE.  if (xl > yu or xu < yl) or  .NOT. (xl <= xu and yl <=yu)
```

POSSIBLY TRUE relationals

If an empty interval is an operand of a POSSIBLY TRUE relational operator then the result is FALSE.

The result of the .PNE. operator is also FALSE because an empty interval is certainly not equal to anything else.

Additional (in comparison to [2]) checks `(xl <= xu and yl <=yu)` assure that the result is FALSE if an empty interval is an operand.

```
X.PLT.Y       .TRUE.  if xl < yu   and xl <= xu and yl <=yu

X.PLE.Y       .TRUE.  if xl <= yu  and xl <= xu and yl <=yu

X.PGT.Y       .TRUE.  if xu > yl   and xl <= xu and yl <=yu

X.PGE.Y       .TRUE.  if xu >= yl  and xl <= xu and yl <=yu
```

The algorithms of .PEQ. and .PNE. operators remain unchanged.

```
X.PEQ.Y       .TRUE.  if  xu >= yl and xl <=yu

X.PNE.Y       .TRUE.  if  xu > yl and xl < yu
```

Equality and inequality of intervals as sets

If an empty interval is an operand of the .SEQ. operator then the result is FALSE.

```
X.SEQ.Y       .TRUE. if xl=yl and xu=yu
```

If an empty interval is an operand of the .SNE. operator then the result is TRUE.

In contrast to the algorithm in [2] the .SNE. operator should not be defined comparing the bounds of interval operands like $(xl \neq yl)$ .OR. $(xu \neq yu)$ because depending on the implementation, NaN may compare as FALSE with anything else and we may not get the desired TRUE result if an empty interval is an operand.

For example

> [1,2] .SNE. [1,NaN] must be TRUE,
> but $(1 \neq 1)$ .OR. $(2 \neq NaN)$ may be FALSE

Therefore the algorithm should use the negation of the .SEQ. operator

```
X.SNE.Y       .TRUE. if .NOT. ( xl=yl and xu=yu)
```

5 Special interval functions

If an empty interval is an operand of the following functions then the result is NaN. The width of an empty interval is also NaN.

```
R = MID(X)    Midpoint of X

R = WID(X)    R <-- xu - xl

R = MAG(X)    R <-- max { |xl|, |xu| }  "Magnitude"


                   | min { |xl|, |xu| } if .NOT.(0.IN.X)
R = MIG(X)    R <--|
                   |  0,  otherwise.

              "Mignitude"
```

If an empty interval is the operand of the ABS function then the result is an empty interval.

```
                     |
Z = ABS(X)     Z <-- | [min{|x|}, max{|x|}]
                     |  x.IN.X    x.IN.X
               Range of absolute value
```

If an empty interval is an operand of the following functions then the result is an empty interval.

```
Z = MAX(X,Y)  Z <-- [max {xl,yl}, max {xu,yu}]

              Range of maximum
              MAX shall be extended analogously
              for more than two
              arguments.

Z = MIN(X,Y)  Z <-- [min {xl,yl}, min {xu,yu}]

              Range of minimum
              MIN shall be extended analogously
              for more than two
              arguments.
```

If an empty interval is the operand of NDIGITS function then it's result is zero.

```
N = NDIGITS(X) Number of leading decimal digits that are the same in
              xl and xu. n digits shall be counted as the same if
              rounding xl to the nearest decimal number with n
              significant digits gives the same result as rounding
              xu to the nearest decimal number with n significant
              digits.
```

6 Interval versions of the intrinsic functions

All Fortran intrinsic functions that accept real data shall also accept interval data.

All functions shall return enclosures of the range.

The interval argument of a function is intersected with the real-valued domain of that function

Eg. $\sqrt{(X)}$ is implicitly interpreted as $\sqrt{(X \cap [0, \infty))}$.

The result is NaN if the intersection is empty. See also [3].

References

[1] ANSI/IEEE 754-1985 Standard for Binary Floating-Point Arithmetic, Institute of Electrical and Electronics Engineers, New York, 1985.

[2] ANSI X3J3 1996-156, Interval Arithmetic—The Data Type and Low-Level Operations, 1996.

[3] X3J3/97-141, ISO/IEC JTC1/SC22/WG5 - N1231, Proposed Syntax – Exceptions for Interval Intrinsic Functions.

[4] Priest D., *Handling IEEE 754 Invalid Operation Exceptions in Real Interval Arithmetic*, Manuscript, 1997.

[5] Popova, E., *Interval Operations Involving NaNs*, in G. Alefeld and A. Frommer, Eds., *Scientific Computing and Validated Numerics: Proceedings of SCAN-95*, Akademie-Verlag, Berlin, 1996.