SECOND NOTE ON BASIC INTERVAL ARITHMETIC FOR IEEE754R

ISL WORK NOTE WN13B

JOHN PRYCE, GEORGE CORLISS, BAKER KEARFOTT, NED NEDIALKOV,
AND SPENCER SMITH
INTERVAL SUBROUTINE LIBRARY TEAM

Contents

1. Context: The Kirchner–Kulisch interval hardware paper

The IFIP Working Group 2.5 on Numerical Software (IFIPWG2.5) wrote on 5th September 2007 to the IEEE Standards Committee concerned with revising the IEEE Floating-Point Arithmetic Standards 754 and 854 (IEEE754R). The letter expressed the unanimous request of IFIPWG2.5 that the following requirement be included in the future computer arithmetic standard:

> *For the data format double precision, interval arithmetic should be made available at the speed of simple floating-point arithmetic.*

IEEE754R (we believe) welcomed this development. They had before them a document defining interval arithmetic operations but, to be the basis of a standards document, it needed more detail. Members of the Interval Subroutine Library (ISL) team were asked to comment, in an email from Ulrich Kulisch that enclosed one from Jim Demmel to Van Snyder raising the issue.

The document is Kirchner and Kulisch's paper [1],[1] called KK below. We now give a response. Demmel's concerns are that:

1. It is hard to get consensus on the definitions of exceptions for intervals — every case needs to be defined, including meaningless operands with a `NaN` as an "endpoint".

Before covering this, we think, one needs to resolve the following:

2. The operation tables, KK page 3, *even for valid inputs*, have cases where the interval operation $C = A \bullet B$ as given will give a C with a `NaN` as an endpoint.

KK's primary aim is to show minimal extra hardware complexity is needed to implement interval arithmetic. But a fundamental issue still needs to be addressed:

[1]It took us some time to find this out.

Dagstuhl Seminar Proceedings 08021
Numerical Validation in Current Hardware Architectures
http://drops.dagstuhl.de/opus/volltexte/2008/1451

3. KK does not state its underlying mathematical interval model. All commonly used models agree on the simple cases (bounded intervals input and output) — the differences is "at the edges". The operations given do not entirely agree either with the Brönnimann–Melquiond–Pion proposal [2], or with any system based on csets as described e.g. in [3].

Section 2 lists these `NaN` cases of item 2. Section 3 compares, in each such case, the KK result with that of: (a) KKM — what we believe is the mathematical model of KK; (b) the `n2137` model; and (c) Cset0, the most straightforward cset model. Section 4 addresses the issue of exceptions. Section 5 gives recommendations.

2. NaN cases in the KK operation tables

We think Tables 1, 2, 3, 4 give a complete list of the `NaN` cases.[2]

Table 1. Addition. Subtraction has similar cases.

Case	A + B = C	When?	Note
add	$[-\infty,-\infty] + [b_1,+\infty] = [-\infty,\texttt{NaN}]$	if $b_1 < +\infty$	
	$[+\infty,+\infty] + [-\infty,b_2] = [\texttt{NaN},+\infty]$	if $b_2 > -\infty$	
	$[+\infty,+\infty] + [-\infty,-\infty] = [\texttt{NaN},\texttt{NaN}]$		
	Three similar cases with A, B interchanged		

Table 2. Division, $0 \notin B$, KK's cases 1–6 are read by rows.

Case	(definition)	$A \div B = C$	When?
div1	$a_1 \geq 0,\ b_1 > 0$	$[a_1,+\infty] \div [+\infty,+\infty] = [0,\texttt{NaN}]$	if $a_1 < +\infty$
		$[+\infty,+\infty] \div [b_1,+\infty] = [\texttt{NaN},+\infty]$	if $b_1 < +\infty$
		$[+\infty,+\infty] \div [+\infty,+\infty] = [\texttt{NaN},\texttt{NaN}]$	
div2	$a_1 \geq 0,\ b_2 < 0$	3 cases like div1.	
div3	$a_1 < 0 \leq a_2, b_1 > 0$	$[-\infty,a_2] \div [+\infty,+\infty] = [\texttt{NaN},0]$	if $a_2 < +\infty$
		$[a_1,+\infty] \div [+\infty,+\infty] = [0,\texttt{NaN}]$	if $a_1 > -\infty$
		$[-\infty,+\infty] \div [+\infty,+\infty] = [\texttt{NaN},\texttt{NaN}]$	
div4	$a_1 < 0 \leq a_2, b_2 < 0$	3 cases like div3.	
div5	$a_2 < 0,\ b_1 > 0$	3 cases like div1.	
div6	$a_2 < 0,\ b_2 < 0$	3 cases like div1.	

Table 3. Division, $0 \in B$, KK's cases 1–12 are read by rows.

Case	(definition)	$A \div B = C$
divz1	$a_2 < 0,\ b_1 = b_2 = 0$	
		Note: Not all models return empty here.
divz2	$a_2 < 0,\ b_1 < b_2 = 0$	$[-\infty,-\infty] \div [-\infty,0] = [\texttt{NaN},+\infty]$
divz3	$a_2 < 0,\ b_1 < 0 < b_2$	$[-\infty,-\infty] \div [-\infty,+\infty] = [\texttt{NaN},\texttt{NaN}]$
divz4	$a_2 < 0,\ 0 = b_1 < b_2$	$[-\infty,-\infty] \div [0,+\infty] = [-\infty,\texttt{NaN}]$
divz5	$a_1 \leq 0 \leq a_2,\ b_1 = b_2 = 0$	
		Note: Not all models return $[-\infty,+\infty]$ here.
divz9	$a_2 > 0,\ b_1 = b_2 = 0$	like divz1.
divz10	$a_2 > 0,\ b_1 < b_2 = 0$	like divz4.
divz11	$a_2 > 0,\ b_1 < 0 < b_2$	like divz3.
divz12	$a_2 > 0,\ 0 = b_1 < b_2$	like divz2.

[2]Multiplication comes last because it needed to go on a landscape page.

TABLE 4. Multiplication. KK's cases 1–9 are read by rows. A * marks cases that do *not* involve singleton infinite intervals.

Case	(definition)		A	$\times$	B	$=$	C	When?
mul1	$a_1 \geq 0,\ b_1 \geq 0$		$[0, a_2]$	$\times$	$[+\infty, +\infty]$	$=$	$[\mathtt{NaN}, +\infty]$	if $a_2 > 0$
		*	$[0, 0]$	$\times$	$[b_1, +\infty]$	$=$	$[0, \mathtt{NaN}]$	if $b_1 < +\infty$
			$[0, 0]$	$\times$	$[+\infty, +\infty]$	$=$	$[\mathtt{NaN}, \mathtt{NaN}]$	
mul2	$a_1 \geq 0,\ b_1 < 0 \leq b_2$	*	$[0, 0]$	$\times$	$[-\infty, b_2]$	$=$	$[\mathtt{NaN}, 0]$	if $b_2 < +\infty$
		*	$[0, 0]$	$\times$	$[-\infty, +\infty]$	$=$	$[\mathtt{NaN}, \mathtt{NaN}]$	
		*	$[a_1, +\infty]$	$\times$	$[b_1, 0]$	$=$	$[-\infty, \mathtt{NaN}]$	
mul3	$a_1 \geq 0,\ b_2 < 0$	*	$[0, 0]$	$\times$	$[-\infty, b_2]$	$=$	$[\mathtt{NaN}, 0]]$	if $b_2 > -\infty$
			$[0, 0]$	$\times$	$[-\infty, -\infty]$	$=$	$[\mathtt{NaN}, \mathtt{NaN}]$	
mul4	$a_1 < 0 \leq a_2,\ b_1 \geq 0$	*	$[-\infty, a_2]$	$\times$	$[0, 0]$	$=$	$[\mathtt{NaN}, 0]$	if $a_2 < +\infty$
		*	$[-\infty, +\infty]$	$\times$	$[0, 0]$	$=$	$[\mathtt{NaN}, \mathtt{NaN}]$	
mul5	$a_1{<}0{\leq}a_2, b_1{<}0{\leq}b_2$	*	$[-\infty, a_2]$	$\times$	$[b_1, 0]$	$=$	$[\min(\mathtt{NaN}, a_2 \mathbin{\overset{\smile}{\times}} b_1),$ $\max(+\infty, 0)]$	
						$=$	$[a_2 \mathbin{\overset{\smile}{\times}} b_1, +\infty]$	if $a_2 < +\infty$
		*	$[-\infty, +\infty]$	$\times$	$[b_1, 0]$	$=$	$[\min(\mathtt{NaN}, -\infty),$ $\max(+\infty, \mathtt{NaN})]$	
						$=$	$[-\infty, +\infty]$	
		*	$[a_1, 0]$	$\times$	$[-\infty, b_2]$	$=$	$[\min(a_1 \mathbin{\overset{\smile}{\times}} b_2, \mathtt{NaN}),$ $\max(+\infty, 0)]$	
						$=$	$[a_1 \mathbin{\overset{\smile}{\times}} b_2, +\infty]$	if $b_2 < +\infty$
		*	$[a_1, 0]$	$\times$	$[-\infty, +\infty]$	$=$	$[\min(-\infty, \mathtt{NaN}),$ $\max(+\infty, \mathtt{NaN})]$	
						$=$	$[-\infty, +\infty]$	
mul6	$a_1 < 0 \leq a_2,\ b_2 < 0$	*	$[a_1, 0]$	$\times$	$[-\infty, b_2]$	$=$	$[\mathtt{NaN}, +\infty]$	
mul7	$a_2 < 0,\ b_1 \geq 0$	*	$[-\infty, a_2]$	$\times$	$[0, 0]$	$=$	$[\mathtt{NaN}, 0]$	if $a_2 > -\infty$
			$[-\infty, -\infty]$	$\times$	$[0, 0]$	$=$	$[\mathtt{NaN}, \mathtt{NaN}]$	
mul8	$a_2 < 0,\ b_1 < 0 \leq b_2$	*	$[-\infty, a_2]$	$\times$	$[b_1, 0]$	$=$	$[\mathtt{NaN}, +\infty]$	
mul9	$a_2 < 0,\ b_2 < 0$		no strange cases					

3. Comparison of KK, n2137 and cset0

Tables 5, 6, 7, 8 compare the results from the three models, but *only* on the cases listed in Section 2. There are several other cases where the models give different results.

3.1. **Definitions.** In all three models an interval result $A \bullet B$ is found by computing a "raw set result" and replacing this by *the smallest model interval containing it*.

In the `n2137` model the number system is the reals $\mathbb{R}$, and the intervals are all *nonempty* $[a_1, a_2] = \{x \in \mathbb{R} \mid a_1 \le x \le a_2\}$ where $-\infty \le a_1 \le a_2 \le +\infty$ ("nonempty" forbids $a_1 = a_2 = \pm\infty$), plus the empty set. The raw set result $A \bullet B$ is $\{a \bullet b \mid a \in A \text{ and } b \in B\}$. Division $x \div y$ is a function in the usual way so $\{1\} \div \{0\}$ and $\{0\} \div \{0\}$ are both empty.

KKM is what we believe is the model underlying the KK operations. The number system and intervals are as in `n2137`. The operations are the same except (for point values) $a \div b$ means "any solution c of $bc = a$" — the raw set result $A \div B$ is the set of c such that $bc = a$ for some $a \in A$, $b \in B$. Thus $\{1\} \div \{0\}$ is empty and $\{0\} \div \{0\}$ is all of $\mathbb{R}$.

In the Cset0 model used here, the number system is the extended reals $\mathbb{R}^* = \mathbb{R} \cup \{-\infty, +\infty\}$. The intervals are all $[a_1, a_2] = \{x \in \mathbb{R}^* \mid a_1 \le x \le a_2\}$ where $-\infty \le a_1 \le a_2 \le +\infty$, plus the empty set. The raw set result $A \bullet B$ is the set of all limits $\lim_r a_r \bullet b_r$ where (a_r) is a sequence converging to some point of A, (b_r) is similar, and $a_r \bullet b_r$ is defined for all r in the normal way. In the present context this means the a_r and b_r are finite and, when $\bullet$ is $\div$, all $b_r \ne 0$. Thus $\{1\} \div \{0\}$ is $\{-\infty, +\infty\}$ and $\{0\} \div \{0\}$ is all of $\mathbb{R}^*$.

In all these models, there is only one zero: IEEE `-0` and `+0` must be treated as identical.

At the implementation level, an `interval` (typewriter font) will mean a 128-bit field `x` made up of two 64-bit `double` fields `x1`, `x2`. We write `x = [x1, x2]`. An `interval` value is *valid* if it denotes a (possibly empty) interval in the chosen model, otherwise *invalid*.

3.2. **KK's division by zero.** The mantra "smallest model interval containing ..." is relaxed for the case of $A \div B$ where $0 \in B$. In this case the raw set result can be a wraparound (exterior) interval of the form $[-\infty, c_2] \cup [c_1, +\infty]$. KK allows the return of such an interval, encoded as $[c_1, c_2]$ where $c_1 > c_2$. It must be split into its two parts before being used in further arithmetic operations.

This looks a nice design decision so we have assumed it for all three models. For `n2137`, the only one with a concrete proposal, such a feature was requested but has not made it into the current text as far as we can see. For Cset0, it answers some criticisms that have been made of returning uselessly wide bounds. E.g. without it, $1/[0, 1]$ is $[1, +\infty]$ in `n2137` but $[-\infty, +\infty]$ in Cset0. With it, one still has $[1, +\infty]$ in `n2137` but $[-\infty] \cup [1, +\infty]$ in Cset0.

3.3. **Discussion.** First, three mathematically sensible interval models, all with support from respected groups in the interval community, behave significantly differently "at the edges" of arithmetic.

Second, it is not hard to see that for all cases, the KK formulas deliver the cset result automatically if IEEE754 arithmetic is changed so that *when rounding is downward or upward, non-`NaN` inputs never return `NaN`.* Except for division with $0 \in B$, they are to return:

> *rounding downwards:* the minimum (infimum) of the cset result;
> *rounding upwards:* the maximum (supremum) of the cset result.

For instance $+\infty \mathbin{\underset{\smile}{\div}} +\infty = 0$, $+\infty \mathbin{\widehat{\div}} +\infty = +\infty$.

For division with $0 \in B$, the situation is interesting. Using KK's proposed support for wraparound intervals in this special case, one gets the correct cset result by doing two things. First, for division by zero make a wraparound version of the above downwards/upwards rule:

$$x \mathbin{\underset{\smile}{\div}} 0 = +\infty, \quad x \mathbin{\widehat{\div}} 0 = -\infty \qquad \text{if } x \ne 0; \text{ but}$$
$$0 \mathbin{\underset{\smile}{\div}} 0 = -\infty, \quad 0 \mathbin{\widehat{\div}} 0 = +\infty$$

Second, simplify KK's table for division with $0 \in B$ to the following:

	$B = [b_1, b_2]$ arbitrary
$a_2 < 0$	$[a_2 \mathbin{\underset{\smile}{\div}} b_1, a_2 \mathbin{\widehat{\div}} b_2]$
$a_1 \le 0 \le a_2$	$[-\infty, +\infty]$
$a_1 > 0$	$[a_1 \mathbin{\underset{\smile}{\div}} b_2, a_1 \mathbin{\widehat{\div}} b_1]$

For this case, the cset result always equals (KK result) $\cup \{-\infty, +\infty\}$; it is simple to verify the above table gives this result.

We suppose there is little chance of influencing IEEE754's handling of rounding at this late stage, but we believe changing it as above leads to simpler logic in the proposed hardware implementation.

TABLE 5. Addition. Subtraction has similar cases. All involve singleton infinite intervals.

Case	A + B =	KK	KKM, `n2137`	Cset0	When?
add	$[-\infty, -\infty] + [b_1, +\infty] =$	$[-\infty, \mathtt{NaN}]$	illegal	$[-\infty, +\infty]$	$(b_1 > -\infty)$
	$[+\infty, +\infty] + [-\infty, b_2] =$	$[\mathtt{NaN}, +\infty]$	illegal	$[-\infty, +\infty]$	$(b_2 < +\infty)$
	$[+\infty, +\infty] + [-\infty, -\infty] =$	$[\mathtt{NaN}, \mathtt{NaN}]$	illegal	$[-\infty, +\infty]$	
	Three similar cases with A, B interchanged				

TABLE 6. Division, $0 \notin B$, KK's cases 1–6 are read by rows. All involve singleton infinite intervals.

	A ÷ B =	KK	KKM, `n2137`	Cset0	When?
div1	$[a_1, +\infty] \div [+\infty, +\infty] =$	$[0, \mathtt{NaN}]$	illegal	$[0, +\infty]$	
	$[+\infty, +\infty] \div [b_1, +\infty] =$	$[\mathtt{NaN}, +\infty]$	illegal	$[0, +\infty]$	
	$[+\infty, +\infty] \div [+\infty, +\infty] =$	$[\mathtt{NaN}, \mathtt{NaN}]$	illegal	$[0, +\infty]$	
div2	3 cases like div1.				
div3	$[-\infty, a_2] \div [+\infty, +\infty] =$	$[\mathtt{NaN}, 0]$	illegal	$[-\infty, 0]$	$(a_2 < +\infty)$
	$[a_1, +\infty] \div [+\infty, +\infty] =$	$[0, \mathtt{NaN}]$	illegal	$[0, +\infty]$	$(a_1 > -\infty)$
	$[-\infty, +\infty] \div [+\infty, +\infty] =$	$[\mathtt{NaN}, \mathtt{NaN}]$	illegal	$[-\infty, +\infty]$	
div4	3 cases like div3.				
div5	3 cases like div1.				
div6	3 cases like div1.				

TABLE 7. Division, $0 \in B$, KK's cases 1–12 are read by rows. † marks Cset0 results that wrap around, so can be handled by KK's proposed wraparound mechanism. Note $[+\infty, -\infty]$ is the union of the two infinite singleton intervals $[-\infty, -\infty]$ and $[+\infty, +\infty]$.

Case		A ÷ B =	KK	KKM	`n2137`	Cset0
divz1	$a_2 < 0,\ b_1 = b_2 = 0$	$[a_1, a_2] \div [0, 0] =$	$[+\mathtt{NaN}, -\mathtt{NaN}]$	empty	empty	$[+\infty, -\infty]$ †
divz2	$a_2 < 0,\ b_1 < b_2 = 0$	$[-\infty, -\infty] \div [-\infty, 0] =$	$[\mathtt{NaN}, +\infty]$	illegal	illegal	$[0, -\infty]$ †
divz3	$a_2 < 0,\ b_1 < 0 < b_2$	$[-\infty, -\infty] \div [-\infty, +\infty] =$	$[\mathtt{NaN}, \mathtt{NaN}]$	illegal	illegal	$[-\infty, +\infty]$
divz4	$a_2 < 0,\ 0 = b_1 < b_2$	$[-\infty, -\infty] \div [0, +\infty] =$	$[-\infty, \mathtt{NaN}]$	illegal	illegal	$[+\infty, 0]$ †
divz5	$a_1 \le 0 \le a_2, b_1 = b_2 = 0$	$[a_1, a_2] \div [0, 0] =$	$[-\infty, +\infty]$	$[-\infty, +\infty]$	empty	$[-\infty, +\infty]$
divz9	$a_2 > 0,\ b_1 = b_2 = 0$	like divz1.				
divz10	$a_2 > 0,\ b_1 < b_2 = 0$	like divz4.				
divz11	$a_2 > 0,\ b_1 < 0 < b_2$	like divz3.				
divz12	$a_2 > 0,\ 0 = b_1 < b_2$	like divz2.				

TABLE 8. Multiplication. KK's cases 1–9 are read by rows. A * marks cases that do *not* involve singleton infinite intervals. The Cset0 result is $[-\infty, +\infty]$ in each of these cases.

Case			A	×	B	KK	KKM, `n2137`	Cset0	When?
mul1	$a_1 \geq 0,\ b_1 \geq 0$		$[0, a_2]$	×	$[+\infty, +\infty]$	$[\texttt{NaN}, +\infty]$	illegal	$[-\infty, +\infty]$	if $a_2 > 0$
		*	$[0, 0]$	×	$[b_1, +\infty]$	$[0, \texttt{NaN}]$	$[0, 0]$	$[-\infty, +\infty]$	if $b_1 < +\infty$
			$[0, 0]$	×	$[+\infty, +\infty]$	$[\texttt{NaN}, \texttt{NaN}]$	illegal	$[-\infty, +\infty]$	
mul2	$a_1 \geq 0,\ b_1 < 0 \leq b_2$	*	$[0, 0]$	×	$[-\infty, b_2]$	$[\texttt{NaN}, 0]$	$[0, 0]$	$[-\infty, +\infty]$	if $b_2 < +\infty$
		*	$[0, 0]$	×	$[-\infty, +\infty]$	$[\texttt{NaN}, \texttt{NaN}]$	$[0, 0]$	$[-\infty, +\infty]$	
		*	$[a_1, +\infty]$	×	$[b_1, 0]$	$[-\infty, \texttt{NaN}]$	$[-\infty, 0]$	$[-\infty, +\infty]$	
mul3	$a_1 \geq 0,\ b_2 < 0$	*	$[0, 0]$	×	$[-\infty, b_2]$	$[\texttt{NaN}, 0]]$	$[0, 0]$	$[-\infty, +\infty]$	if $b_2 > -\infty$
			$[0, 0]$	×	$[-\infty, -\infty]$	$[\texttt{NaN}, \texttt{NaN}]$	illegal	$[-\infty, +\infty]$	
mul4	$a_1 < 0 \leq a_2,\ b_1 \geq 0$	*	$[-\infty, a_2]$	×	$[0, 0]$	$[\texttt{NaN}, 0]$	$[0, 0]$	$[-\infty, +\infty]$	if $a_2 < -\infty$
		*	$[-\infty, +\infty]$	×	$[0, 0]$	$[\texttt{NaN}, \texttt{NaN}]$	$[0, 0]$	$[-\infty, +\infty]$	
		*	$[a_1, 0]$	×	$[b_1, +\infty]$	$[\texttt{NaN}, \texttt{NaN}]$	$[0, 0]$	$[-\infty, +\infty]$	
mul5	$a_1 < 0 \leq a_2,\ b_1 < 0 \leq b_2$	*	$[-\infty, a_2]$	×	$[b_1, 0]$	$[a_2 \underset{\smile}{\times} b_1, +\infty]$	$[a_2 \underset{\smile}{\times} b_1, +\infty]$	$[-\infty, +\infty]$	if $a_2 < +\infty$
		*	$[-\infty, +\infty]$	×	$[b_1, 0]$	$[-\infty, +\infty]$	$[-\infty, +\infty]$	$[-\infty, +\infty]$	
		*	$[a_1, 0]$	×	$[-\infty, b_2]$	$[a_1 \underset{\smile}{\times} b_2, +\infty]$	$[a_1 \underset{\smile}{\times} b_2, +\infty]$	$[-\infty, +\infty]$	if $b_2 < +\infty$
		*	$[a_1, 0]$	×	$[-\infty, +\infty]$	$[-\infty, +\infty]$	$[-\infty, +\infty]$	$[-\infty, +\infty]$	
mul6	$a_1 < 0 \leq a_2,\ b_2 < 0$	*	$[a_1, 0]$	×	$[-\infty, b_2]$	$[\texttt{NaN}, +\infty]$	$[0, +\infty]$	$[-\infty, +\infty]$	
mul7	$a_2 < 0,\ b_1 \geq 0$	*	$[-\infty, a_2]$	×	$[0, 0]$	$[\texttt{NaN}, 0]$	$[0, 0]$	$[-\infty, +\infty]$	if $a_2 > -\infty$
			$[-\infty, -\infty]$	×	$[0, 0]$	$[\texttt{NaN}, \texttt{NaN}]$	illegal	$[-\infty, +\infty]$	
mul8	$a_2 < 0,\ b_1 < 0 \leq b_2$	*	$[-\infty, a_2]$	×	$[b_1, 0]$	$[\texttt{NaN}, +\infty]$	$[0, +\infty]$	$[-\infty, +\infty]$	
mul9	$a_2 < 0,\ b_2 < 0$		no strange cases						

4. Exceptions/Flags

4.1. **Invalid and NaI.** Whichever of the three models (or another) is chosen, the result of any arithmetic operation on two valid intervals is another valid interval. Therefore arithmetic operations raise `INVALID` if, and only if, given an invalid interval as input.

`n2137` requires a special `interval` value that we here call "Not an Interval", `NaI`. It could be a class of values, carrying a "payload" like `NaN`. It is useful in various ways for making programming more secure, so ISL agree it should be in the standard.

4.2. **Discontinuous.** It is essential to have a `DISCTS` flag, raised when the inputs to an operation $\bullet$ are outside the domain where $\bullet$ is defined and continuous. If one does not have it, the many interval algorithms that rely on Brouwer's Theorem (more-or-less anything to do with differential equations) are condemned to run an order of magnitude slower, and are harder to code.

In KKM and `n2137`, based on $\mathbb{R}$, the only place where `DISCTS` is raised is (any interval arithmetic operation that implies) division by zero. In Cset0, based on $\mathbb{R}^*$, all the operations that give `NaN` in IEEE arithmetic *also* raise `DISCTS`. That is, in addition to $x \div 0$, anything equivalent to $\infty - \infty$, $0 \times \infty$, $\infty \div \infty$.

Mathematically, this is because a function is discontinuous at a point if, and only if, its cset value there is a set of more than one point; this is exactly when IEEE754 specifies `NaN`.

4.3. **Inexact, Underflow, Overflow.** We presume the main rationale of these in interval work is for fine-tuning of algorithms, detecting opportunities of special action, e.g. to get tighter bounds. If this is so, a reasonable rule is that an interval arithmetic operation should raise `INEXACT`, `UNDERFLOW` or `OVERFLOW` if, and only if, at least one of the floating point operations in its implementation does so. This raises a problem with KK's multiplication case 5, "mul5", which contains sub-cases due to using max() and min().

Example 1. Let m and M be the smallest positive normalized and largest finite floating point numbers. Then $[-m, M] \times [-m, M]$ is evaluated as

$$[\min(-m \mathbin{\underset{\smile}{\times}} M, M \mathbin{\underset{\smile}{\times}} (-m)),\ \max(-m \mathbin{\widehat{\times}} (-m), M \mathbin{\widehat{\times}} M)] = [-4, +\infty].$$

My inclination is that this operation should only raise `OVERFLOW`, although one of the operations done was $(-m) \mathbin{\widehat{\times}} (-m)$ which returns the smallest positive denormal (because rounding up) and raises `UNDERFLOW`.

Example 2. With M as above, $[-M, 0] \times [-\infty, M]$ is evaluated as $[\min(-M \mathbin{\underset{\smile}{\times}} M, 0 \mathbin{\underset{\smile}{\times}} (-\infty)), \max(-M \mathbin{\widehat{\times}} (-\infty), 0 \mathbin{\widehat{\times}} M)]$. The right endpoint is $+\infty$ with no flags raised. The left endpoint is problematic. With current IEEE754 arithmetic it is $\min(-\infty, \texttt{NaN})$ with `OVERFLOW` from the $-\infty$. With our suggested change to arithmetic it is $\min(-\infty, -\infty)$ with `OVERFLOW` from the first $-\infty$, not the second. The correct overall result is $[-\infty, +\infty]$ in all three models. In the Cset0 model we incline to *not* raising `OVERFLOW` for the overall operation; in KKM and `n2137`, we do not know.

Example 3. If a is the nearest `double` number to $1/3$, then $[-a, 1] \times [-3, 9]$ is evaluated as $[\min(-a \mathbin{\underset{\smile}{\times}} 9, 1 \mathbin{\underset{\smile}{\times}} (-3)), \max((-a) \mathbin{\widehat{\times}} (-3), 1 \mathbin{\widehat{\times}} 9)]$. If we have it right, $-a \mathbin{\underset{\smile}{\times}} 9$ is -3 exactly so the answer is $[-3, 9]$ exactly. The right endpoint did not raise `INEXACT`, but the left endpoint came from two equal answers, $-a \mathbin{\underset{\smile}{\times}} 9$ which raised `INEXACT` and $1 \mathbin{\underset{\smile}{\times}} (-3)$ which did not. My inclination is that the overall operation should *not* raise `INEXACT`.

Here is a possible rule for case mul5, where `FLAG` is `INEXACT`, `UNDERFLOW` or `OVERFLOW`:

> Do *not* raise `FLAG`, if and only if it is possible to choose c_1 as one of $a_1 \mathbin{\underset{\smile}{\times}} b_2$ and $a_2 \mathbin{\underset{\smile}{\times}} b_1$, and c_2 as one of $a_1 \mathbin{\widehat{\times}} b_1$ and $a_2 \mathbin{\widehat{\times}} b_2$, so as to give the correct result $[c_1, c_2]$ and not raise `FLAG`.

5. Recommendations

We recall that IEEE754 does not specify whether the standard is implemented in hardware, software, or a mix of the two.

These recommendations are in the context of the basic structure offered by KK. They are the considered view of the ISL team.

(1) We suggest the IEEE754R Committee choose *one* of KKM, `n2137` or Cset0 as the model mandated by the standard.
(2) The standard should aim, as far as practical, to make it easy to implement other models on top of the chosen model without significant loss in performance.
(3) There shall be a (necessarily invalid) `interval` value denoting "Not an Interval", `NaI`.
(4) There shall an `interval` value `empty` denoting the empty set. (The standard should not mandate that it be KK's value `[+NaN, -NaN]`.)
(5) `NaI`, `empty` and any other special values shall be different (i) from each other, and (ii) from all `[x1, x2]` where `x1` and `x2` are (finite or infinite) `double` numeric values. (This supports recommendation 2 by keeping all numeric `[x1, x2]` free for use by models that use, e.g., wraparound intervals.)
(6) There shall be a `DISCTS` flag, as defined in Subsection 4.2.
(7) An interval arithmetic operation shall raise `INVALID` if, and only if, at least one of its inputs is an invalid interval. It shall then return `NaI` as its result.
(8) An interval arithmetic operation shall raise `INEXACT`, `UNDERFLOW` or `OVERFLOW` if, and only if, at least one of the floating point operations in its implementation does so, *except* in KK's case mul5, where we suggest the rule in Subsection 4.3 but further discussion is needed.
(9) We suggest an *interval validate* operation: `validate(x)` shall examine an `interval` value `x = [x1, x2]`, and raise `INVALID` if, and only if, it does not represent a valid interval in the chosen model.
(10) There shall be an operation to split a wraparound interval (e.g., returned by $A \div B$ when $0 \in B$) into its two interval parts. It shall behave as if implemented thus:
    Convert `[c1, c2]` to the pair `[-inf, c2]` and `[c1, +inf]`. Then apply `validate()` to each of these.
    Thus it may raise `INVALID` but not any other flags.

Minor point. This is not directly relevant to the standard, but affects it indirectly. We suggest the cases in KK's multiplication be changed from ($a_1 \geq 0$; $a_1 < 0 \leq a_2$; $a_2 < 0$) to ($a_1 \geq 0$; $a_1 < 0 < a_2$; $a_2 \leq 0$), and similarly for b_1, b_2. We think the resulting symmetry simplifies the logic and we conjecture it will simplify circuit design.

References

[1] Reinhard Kirchner and Ulrich W. Kulisch. *Hardware support for interval arithmetic.* Reliable Computing, Vol 12, Number 3 (June 2006), pp.225-237.
[2] Hervé Brönnimann, Guillaume Melquiond and Sylvain Pion. *A Proposal to add Interval Arithmetic to the C++ Standard Library (revision 2).* Nov 2006. `C++` Standards Committee document N2137.
[3] John D. Pryce and George F. Corliss. *Interval Arithmetic with Containment Sets.* Computing, Vol 78, Number 3 (November 2006), pp.251-276. Publ. Springer Verlag. Also available at `http://www.cas.mcmaster.ca/~isl/Publications/IntvlArithCsets.pdf`

Corresponding author: John Pryce
`j.d.pryce@ntlworld.com`
November 18, 2007