Complete Interval Arithmetic and its Implementation on the Computer

Ulrich W. Kulisch
Institut für Angewandte und Numerische Mathematik
Universität Karlsruhe

Abstract: Let $I\mathbb{R}$ be the set of closed and bounded intervals of real numbers. Arithmetic in $I\mathbb{R}$ can be defined via the power set $\mathbb{P}\mathbb{R}$ of real numbers. If divisors containing zero are excluded, arithmetic in $I\mathbb{R}$ is an algebraically closed subset of the arithmetic in $\mathbb{P}\mathbb{R}$, i.e., an operation in $I\mathbb{R}$ performed in $\mathbb{P}\mathbb{R}$ gives a result that is in $I\mathbb{R}$. Arithmetic in $\mathbb{P}\mathbb{R}$ also allows division by an interval that contains zero. Such division results in closed intervals of real numbers which, however, are no longer bounded. The union of the set $I\mathbb{R}$ with these new intervals is denoted by $(I\mathbb{R})$. This paper shows that arithmetic operations can be extended to all elements of the set $(I\mathbb{R})$.

Let $F \subset \mathbb{R}$ denote the set of floating-point numbers. On the computer, arithmetic in $(I\mathbb{R})$ is approximated by arithmetic in the subset (IF) of closed intervals with floating-point bounds. The usual exceptions of floating-point arithmetic like underflow, overflow, division by zero, or invalid operation do not occur in (IF).

Keywords: computer arithmetic, floating-point arithmetic, interval arithmetic, arithmetic standards.

1 Introduction or a Vision of Future Computing

Computers are getting ever faster. On all relevant processors floating-point arithmetic is provided by fast hardware. The time can already be foreseen when the PC will be a teraflops computer. With this tremendous computing power scientific computing will experience a significant shift from floating-point arithmetic toward increased use of interval arithmetic. With very little extra hardware, interval arithmetic can be made as fast as simple floating-point arithmetic [3]. Nearly everything that is needed for fast interval arithmetic is already available on most existing processors, thanks to multimedia applications. What is still missing are the arithmetic operations with the directed roundings. Properly developed, interval arithmetic is a complete and exception-free calculus. The exceptions of floating-point arithmetic like underflow, overflow, division by zero, or operation invalidity do not occur in such interval arithmetic. This will be shown in the following.

For interval evaluation of an algorithm (a sequence of arithmetic operations) in the real number field a theorem by R. E. Moore [7] states that increasing the precision by k digits reduces the error bounds by b^{-k}, i.e., results can always be guaranteed to a number of correct digits by using variable precision interval arithmetic (for details see [1], [9]). Variable length interval arithmetic can be made very fast by an exact dot product and complete arithmetic [4]. There is no way to compute a dot product faster than the exact result. By pipelining, it can be computed in the time the processor needs to read the data, i.e., it comes with utmost speed. Variable length interval arithmetic fully benefits from such speed. No software simulation can go as fast. With operator overloading variable length interval arithmetic is very easy to use.

The tremendous progress in computer technology should be accompanied by extension of the mathematical capacity of the computer. A balanced standard for computer arithmetic should require that the basic components of modern computing (floating-point arithmetic, interval arithmetic, and an exact dot product) be provided by the computer's hardware. See [5].

2 Remarks on Floating-Point Arithmetic

Computing is usually done in the set of real numbers $I\!R$. The real numbers can be defined as a conditionally complete, linearly ordered field. Conditionally complete means that every bounded subset has an infimum and a supremum. Every conditionally ordered set can be completed by joining a least and a greatest element. In case of the real numbers these are called $-\infty$ and $+\infty$. Then $I\!R^* := I\!R \cup \{-\infty\} \cup \{+\infty\}$ is a complete lattice. We remark, however, that the elements $-\infty$ and $+\infty$ are not elements of the field. The cancellation law $a + c = b + c \Rightarrow a = b$, for instance, does not hold for $c = \infty$.

A real number consists of a sign, an integral part, and a fractional part, for instance: $\pm 345.789123 \cdots \in I\!R$. The point may be shifted to any other position if we compensate for this shifting by a corresponding power of b (here $b = 10$). If the point is shifted immediately to the left of the first nonzero digit: $\pm 0.345789123 \cdots \cdot 10^3$ the representation is called normalized. Zero is the only real number that has no such representation. It needs a special encoding. Thus a normalized real number consists of a signed fractional part m (mantissa) and an integer exponent e and we have $0.1 \leq |m| < 1$.

Only subsets of these numbers can be represented on the computer. If the mantissa in truncated after the l^{th} digit and the exponent is limited by $e_{min} < e < e_{max}$ one speaks of a floating-point number. Let F denote the set of floating-point numbers and $F^* := F \cup \{-\infty\} \cup \{+\infty\}$. [1]

Arithmetic for floating-point numbers may cause exceptions. Well known such exceptions are underflow, overflow, division by zero, or invalid operation. To avoid interruption of program execution in case of an exception the so-called IEEE floating-point arithmetic standard provides additional elements and defines operations for these, for instance, $4/0 := \infty, -4/0 := -\infty, \infty - \infty := NaN, 0 \cdot \infty := NaN, \infty/\infty := NaN, 0/0 := NaN, 1/(-\infty) := -0, (-0.3)/\infty := -0$. It should be clear, however, that $-\infty, +\infty, NaN, -0$,[2] or $+0$ with their operations are not elements of the real number field.

3 Arithmetic for Intervals of $I\!I\!R$ and IF

Interval arithmetic is another arithmetic tool. It solely deals with sets of real numbers. Neither the exceptions of floating-point arithmetic mentioned above nor the measures to deal with them occur or are needed in interval arithmetic. The symbol $I\!I\!R$ usually denotes the set of closed and bounded intervals of $I\!R$. Arithmetic in $I\!I\!R$ can be interpreted as a systematic calculus to deal with inequalities. We assume here that the basic rules for arithmetic in $I\!I\!R$ with zero not in the divisor are known to the reader. It is a fascinating result that, in contrast to floating-point arithmetic, interval arithmetic even on computers can be further developed into a well rounded, exception-free, closed calculus. We briefly sketch this development here.

[1] A rounding maps the complete lattice $I\!R^*$ on the complete sublattice F^*. Both lattices coincide in the least and the greatest element, see [4].

[2] In $I\!R$, 0 is defined as the neutral element of addition. From the assumption that there are two such elements 0 and $0'$ it follows immediately that they are equal: $0 + 0' = 0 = 0'$.

In floating-point arithmetic the crucial operation that leads to the exceptional strategic objects mentioned above is division by zero. So we begin our study of extended interval arithmetic by defining division by an interval that contains zero.

The set $I\mathbb{R}$ is a subset of the power set $\mathbb{P}\mathbb{R}$ (which is the set of all subsets) of real numbers. For $A, B \in \mathbb{P}\mathbb{R}$ arithmetic operations are defined by

$$\bigwedge_{A,B \in \mathbb{P}\mathbb{R}} \quad A \circ B := \{a \circ b \mid a \in A \wedge b \in B\}, \text{ for all } \circ \in \{+, -, \cdot, /\}. \tag{3.1}$$

The following properties are obvious and immediate consequences of this definition:

$$A \subseteq B \wedge C \subseteq D \Rightarrow A \circ C \subseteq B \circ D, \text{ for all } A, B, C, D \in \mathbb{P}\mathbb{R}, \tag{3.2}$$

and in particular

$$a \in A \wedge b \in B \Rightarrow a \circ b \in A \circ B, \text{ for all } A, B \in \mathbb{P}\mathbb{R}. \tag{3.3}$$

Property (3.2) is called *inclusion-isotony* (or *inclusion-monotonicity*). Property (3.3) is called the *inclusion property.* (3.2) and (3.3) are the fundamental properties of interval arithmetic. Under the assumption $0 \notin B$ for division, the intervals of $I\mathbb{R}$ are an algebraically closed subset[3] of the power set $\mathbb{P}\mathbb{R}$, i.e., an operation for intervals of $I\mathbb{R}$ performed in $\mathbb{P}\mathbb{R}$ always delivers an interval of $I\mathbb{R}$.

In real analysis division by zero is not defined. In contrast to this, in interval arithmetic division by an interval that contains zero can be defined in a strict mathematical manner. The result again is a set of real numbers.

In accordance with (3.1) division in $I\mathbb{R}$ is defined by

$$\bigwedge_{A,B \in I\mathbb{R}} \quad A/B := \{a/b \mid a \in A \wedge b \in B\}, \text{ for } 0 \notin B. \tag{3.4}$$

The quotient a/b is defined as the inverse operation of multiplication, i.e., as the solution of the equation $b \cdot x = a$. Thus (3.4) can be written in the form

$$\bigwedge_{A,B \in I\mathbb{R}} \quad A/B := \{x \mid bx = a \wedge a \in A \wedge b \in B\}. \tag{3.5}$$

For $0 \notin B$ (3.4) and (3.5) are equivalent. While in $\mathbb{R}$ division by zero is not defined, the representation of A/B by (3.5) allows definition of the operation and also interpretation of the result for $0 \in B$.

By way of interpreting (3.5) for $A = [a_1, a_2]$ and $B = [b_1, b_2] \in I\mathbb{R}$ with $0 \in B$ the following eight distinct cases can be set out:

1	$0 \in A,$	$0 \in B.$
2	$0 \notin A,$	$B = [0, 0].$
3	$a_1 \le a_2 < 0,$	$b_1 < b_2 = 0.$
4	$a_1 \le a_2 < 0,$	$b_1 < 0 < b_2.$
5	$a_1 \le a_2 < 0,$	$0 = b_1 < b_2.$
6	$0 < a_1 \le a_2,$	$b_1 < b_2 = 0.$
7	$0 < a_1 \le a_2,$	$b_1 < 0 < b_2.$
8	$0 < a_1 \le a_2,$	$0 = b_1 < b_2.$

The list distinguishes the cases $0 \in A$ (case 1) and $0 \notin A$ (cases 2 to 8). Since it is generally assumed that $0 \in B$, these eight cases indeed cover all possibilities. Since every $x \in \mathbb{R}$ fulfills the equation $0 \cdot x = 0$ we obtain in case 1: $A/B = \mathbb{R} = (-\infty, +\infty)$. Here the parentheses indicate that the bounds are not included in the set. In case 2 the set defined by (3.5) consists of all elements which fulfill the

case	$A = [a_1, a_2]$	$B = [b_1, b_2]$	B'	A/B'	A/B
1	$0 \in A$	$0 \in B$			$(-\infty, +\infty)$
2	$0 \notin A$	$B = [0, 0]$			$\varnothing$
3	$a_2 < 0$	$b_1 < b_2 = 0$	$[b_1, (-\epsilon)]$	$[a_2/b_1, a_1/(-\epsilon)]$	$[a_2/b_1, +\infty)$
4	$a_2 < 0$	$b_1 < 0 < b_2$	$[b_1, (-\epsilon)]$	$[a_2/b_1, a_1/(-\epsilon)]$	$(-\infty, a_2/b_2]$
			$\cup\ [\epsilon, b_2]$	$\cup\ [a_1/\epsilon, a_2/b_2]$	$\cup\ [a_2/b_1, +\infty)$
5	$a_2 < 0$	$0 = b_1 < b_2$	$[\epsilon, b_2]$	$[a_1/\epsilon, a_2/b_2]$	$(-\infty, a_2/b_2]$
6	$a_1 > 0$	$b_1 < b_2 = 0$	$[b_1, (-\epsilon)]$	$[a_2/(-\epsilon), a_1/b_1]$	$(-\infty, a_1/b_1]$
7	$a_1 > 0$	$b_1 < 0 < b_2$	$[b_1, (-\epsilon)]$	$[a_2/(-\epsilon), a_1/b_1]$	$(-\infty, a_1/b_1]$
			$\cup\ [\epsilon, b_2]$	$\cup\ [a_1/b_2, a_2/\epsilon]$	$\cup\ [a_1/b_2, +\infty)$
8	$a_1 > 0$	$0 = b_1 < b_2$	$[\epsilon, b_2]$	$[a_1/b_2, a_2/\epsilon]$	$[a_1/b_2, +\infty)$

Table 1: The eight cases of interval division A/B, with $A, B \in I\mathbb{R}$, and $0 \in B$.

equation $0 \cdot x = a$ for $a \in A$. Since $0 \notin A$, there is no real number which fulfills this equation. Thus A/B is the empty set, i.e., $A/B = \varnothing$.

In all other cases $0 \notin A$ also. We have already observed under case 2 that the element 0 in B does not contribute to the solution set. So it can be excluded without changing the set A/B.

So the general rule for computing the set A/B by (3.5) is to remove its zero from the interval B and replace it by a small positive or negative number ϵ as the case may be. The resulting set is denoted by B' and represented in column 4 of Table 1. With this B' the solution set A/B' can now easily be computed by applying the rules for closed and bounded real intervals. The results are shown in column 5 of Table 1. Now the desired result A/B as defined by (3.5) is obtained if in column 5 ϵ tends to zero.

Thus in the cases 3 to 8 the results are obtained by the limit process $A/B = \lim_{\epsilon \to 0} A/B'$. The solution set A/B is shown in the last column of Table 1 for all the eight cases. There, as usual in mathematics, parentheses indicate that the bound is not included in the set. In contrast to this, brackets denote closed interval ends, i.e., the bound is included.

The operands A and B of the division A/B in Table 1 are intervals of $I\mathbb{R}$. The results of the division shown in the last column, however, are no longer intervals of $I\mathbb{R}$. The result is now an element of the power set $\mathbb{P}\mathbb{R}$. With the exception of case 2 the result is now a set which stretches continuously to $-\infty$ or $+\infty$ or both.

In two cases (rows 4 and 7 in Table 1) the result consists of the union of two distinct sets of the form $(-\infty, c_2] \cup [c_1, +\infty)$. These cases can easily be identified by the signs of the bounds of the divisor before the division is executed. For interval multiplication and division a case selection has to be done before the operations are performed anyhow, see [3]. In the two cases (rows 4 and 7 in Table 1) the sign of b_1 is negative and the sign of b_2 is positive.

A basic concept of mathematics is that of a function or mapping. A function consists of a pair (f, D_f). It maps each element x of its domain of definition D_f on a unique element y of the range R_f of f, $f : D_f \to R_f$.

In real analysis division by zero is not defined. Thus a rational function $y = f(x)$ where the denominator is zero for $x = c$ is not defined for $x = c$, i.e., c is not an element of the domain of definition D_f. Since the function $f(x)$ is not defined at

[3]As are the integers of the real numbers for addition, subtraction, and multiplication.

$x = c$ it does not have any value or property there. In this strict mathematical sense, division by an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ is not well posed. For division the set $b_1 < 0 < b_2$ devolves into the two distinct sets $[b_1, 0)$[4] and $(0, b_2]$ and division by the set $b_1 < 0 < b_2$ actually means two divisions. The result of the two divisions consists of the two distinct sets shown in rows 4 and 7 of Table 1. It is highly desirable to perform the two divisions sequentially. Then the two cases (rows 4 and 7) of Table 1 where an operation delivers two distinct results can be eliminated. This simplifies Table 1 considerably. Division by the two sets $[b_1, 0]$ and $[0, b_2]$ is nevertheless defined in Table 1. In these cases only one division has to be performed. The result is a single extended interval. The six remaining cases are shown in Table 2.

case	$A = [a_1, a_2]$	$B = [b_1, b_2]$	A/B
1	$0 \in A$	$0 \in B$	$(-\infty, +\infty)$
2	$0 \notin A$	$B = [0, 0]$	$\varnothing$
3	$a_2 < 0$	$b_1 < b_2 = 0$	$[a_2/b_1, +\infty)$
4	$a_2 < 0$	$0 = b_1 < b_2$	$(-\infty, a_2/b_2]$
5	$a_1 > 0$	$b_1 < b_2 = 0$	$(-\infty, a_1/b_1]$
6	$a_1 > 0$	$0 = b_1 < b_2$	$[a_1/b_2, +\infty)$

Table 2: The six cases of interval division with $A, B \in I\mathbb{R}$, and $0 \in B$.

Thus only four kinds of result come from division by an interval of $I\mathbb{R}$ which contains zero:

$$\varnothing, \quad (-\infty, a], \quad [b, +\infty), \quad \text{and} \quad (-\infty, +\infty). \tag{3.6}$$

We call such elements extended intervals. The union of the set of closed and bounded intervals of $I\mathbb{R}$ with the set of extended intervals is denoted by $(I\mathbb{R})$. The elements of the set $(I\mathbb{R})$ are themselves simply called intervals. $(I\mathbb{R})$ is the set of closed intervals of $\mathbb{R}$. (A subset of $\mathbb{R}$ is called closed if its complement is open.)

Intervals of $I\mathbb{R}$ and of $(I\mathbb{R})$ are sets of real numbers. $-\infty$ and $+\infty$ are not elements of these intervals. It is fascinating that arithmetic operations can be introduced for all elements of the set $(I\mathbb{R})$ in an exception-free manner. This will be shown in the next section.

On a computer only subsets of the real numbers are representable. We assume now that F is the set of floating-point numbers of a given computer. On the computer, arithmetic in $I\mathbb{R}$ is approximated by an arithmetic in IF. IF is the set of closed and bounded intervals with bounds of F. An interval of IF represents the continuous set of real numbers between the floating-point bounds. Arithmetic operations in IF are defined by those in $I\mathbb{R}$ by rounding the result in $I\mathbb{R}$ on the least upper bound in IF with respect to set inclusion as an order relation.

To transform the six cases of division by an interval of $I\mathbb{R}$ which contains zero into computer executable operations we assume now that the operands A and B are floating-point intervals of IF. To obtain a computer representable result we round the result shown in the last column of Table 2 into the least computer representable

[4]Since division by zero does not contribute to the solution set it does not matter whether a paranthesis or bracket is used here.

superset. For finite bounds that is, the lower bound of the result has to be computed with rounding downwards and the upper bound with rounding upwards. Thus on the computer the six cases of division by an interval of IF which contains zero have to be performed as shown in Table 3.

case	$A = [a_1, a_2]$	$B = [b_1, b_2]$	$A \diamondsuit B$
1	$0 \in A$	$0 \in B$	$(-\infty, +\infty)$
2	$0 \notin A$	$B = [0, 0]$	$\varnothing$
3	$a_2 < 0$	$b_1 < b_2 = 0$	$[a_2 \triangledown b_1, +\infty)$
4	$a_2 < 0$	$0 = b_1 < b_2$	$(-\infty, a_2 \triangle b_2]$
5	$a_1 > 0$	$b_1 < b_2 = 0$	$(-\infty, a_1 \triangle b_1]$
6	$a_1 > 0$	$0 = b_1 < b_2$	$[a_1 \triangledown b_2, +\infty)$

Table 3: The six cases of interval division with $A, B \in IF$, and $0 \in B$.

Table 4 shows the same cases as Table 3 in another layout.

	$B = [0, 0]$	$b_1 < b_2 = 0$	$0 = b_1 < b_2$
$a_2 < 0$	$\varnothing$	$[a_2 \triangledown b_1, +\infty)$	$(-\infty, a_2 \triangle b_2]$
$a_1 \leq 0 \leq a_2$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$0 < a_1$	$\varnothing$	$(-\infty, a_1 \triangle b_1]$	$[a_1 \triangledown b_2, +\infty)$

Table 4: The result of the interval division with $A, B \in IF$, and $0 \in B$.

Table 3 and Table 4 display the six distinct cases of interval division $A \diamondsuit B$ with $A, B \in IF$ and $0 \in B$. On the computer the empty interval $\varnothing$ needs a special encoding. We explicitly stress that the symbols $-\infty$, $+\infty$ are used here only to represent the resulting sets. These symbols are not elements of these sets.

In Table 3 and Table 4 and in the following an operator symbol with a $\triangledown$ upon it means an operation performed with rounding downward. Correspondingly an operation with a $\triangle$ upon it means an operation performed with rounding upward.

Table 3 and Table 4 show that division by an interval of IF which contains zero on the computer also leads to extended intervals as shown in (3.6) with $a, b \in F$. The union of the set of closed and bounded intervals of IF with such extended intervals is denoted by (IF). (IF) is the set of closed intervals of real numbers where all finite bounds are elements of F. Except for the empty set, extended intervals also represent continuous sets of real numbers.

Table 2 originated from Table 1 by elimination of rows 4 and 7. Division by an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ actually consists of two divisions by the distinct sets $[b_1, 0)$ and $(0, b_2]$ the result of which again consists of two distinct sets.

In the user's program, however, the two divisions appear as one single operation, as division by an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$. So an arithmetic operation in the user's program delivers two distinct results. This is a totally new situation in computing. But computing certainly is able to cope with this situation.

A solution to the problem would be for the computer to provide a flag for *distinct intervals*. The situation occurs if the divisor is an interval that contains zero as an interior point. In this case the flag would be raised and signaled to the user. The user may then apply a routine of his choice to deal with the situation as is appropriate for his application.[5]

Newton's method reaches its ultimate elegance and strength in the extended interval Newton method. It computes all (single) zeros in a given domain. If a function has several zeros in a given interval its derivative becomes zero in that interval also. Thus Newton's method applied to that interval delivers two distinct sets. This is how the extended interval Newton method separates different zeros from each other. If the method is continued along two separate paths, one for each of the distinct intervals it finally computes all zeros in the given domain. If the method continues with only one of the two distinct sets and ignores the other one it computes an enclosure of only one zero of the given function. If the interval Newton method delivers the empty set, the method has proved that there is no zero in the initial interval.

4 Arithmetic for Intervals of $(I\mathbb{R})$ and (IF)

For the sake of completeness, arithmetic operations now have to be defined for all elements of $(I\mathbb{R})$ and (IF). Since the development of arithmetic operations follows an identical pattern in $(I\mathbb{R})$ and (IF), we skip here the introduction of the arithmetic in $(I\mathbb{R})$ and restrict the consideration to the development of arithmetic in (IF). This is the arithmetic that has to be provided on the computer.

First of all any operation with the empty set is again defined to be the empty set.

The general procedure for defining all other operations follows a continuity principle. Bounds like $-\infty$ and $+\infty$ in the operands A and B are replaced by a very large negative number $(-\Omega)$ and a very large positive number $(+\Omega)$ respectively. Then the basic rules for the arithmetic operations in $I\mathbb{R}$ and IF are applied. In the following tables these rules are repeated and printed in bold letters. In the resulting formulas a very large negative number is then shifted to $-\infty$ and a very large positive number to $+\infty$.

As a short cut for obtaining the resulting rules very simple and well established rules of real analysis like $\infty * x = \infty$ for $x > 0$, $\infty * x = -\infty$ for $x < 0$, $x/\infty = x/-\infty = 0$, $\infty * \infty = \infty$, $(-\infty) * \infty = -\infty$ can be applied together with variants obtained by applying the sign rules and the law of commutativity.

Two situations have to be treated separately. These are the cases shown in rows 1 and 2 of Table 1.

If $0 \in A$ and $0 \in B$ (row 1 of Table 1), the result consists of all the real numbers, i.e., $A/B = (-\infty, +\infty)$. This applies to rows 2, 5, 6 and 8 of Table 9.

If $0 \notin A$ and $B = [0, 0]$ (row 2 of Table 1), the result of the division is the empty set, i.e., $A/B = \emptyset$. This applies to rows 1, 3, 4 and 7 of column 1 of Table 9.

[5]This routine could be: modify the operands and recompute, or continue the computation with one of the sets and ignore the other one, or put one of the sets on a list and continue the computation with the other one, or return the entire set of real numbers $(-\infty, +\infty)$ as result and continue the computation, or stop computing, or any other action.

A somewhat natural solution would be to continue the computation on different processors, one for each interval. But the situation can occur repeatedly. How many processors would we need? Future multicore units will provide a large number of processors. They will suffice for quite a while. A similar situation occurs in global optimization using subdivision. After a certain test several candidates may be left for further investigation.

We outline the complete set of arithmetic operations for interval arithmetic in (IF) that should be provided on the computer in the next section. In summary it can be said that after splitting an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ into two distinct intervals $[b_1, 0)$ and $(0, b_2]$ the result of arithmetic operations for intervals of (IF) always leads to intervals of (IF) again. The reader should prove this assertion by referring to the operations shown in the tables of the following section.

For the development in the preceding sections it was essential to distinguish between parentheses and brackets. If a bound is adjacent to a parenthesis, the bound is not included in the interval; if a bound is adjacent to a bracket, the bound is included in the interval.

In the following tables an operator symbol with a $\bigtriangledown$ upon it means an operation performed with rounding downward. Correspondingly an operation with a $\bigtriangleup$ upon it means an operation performed with rounding upward.

If during a computation in the real number field zero appears as a divisor the computation should be stopped immediately. In floating-point arithmetic the situation is different. Zero may be the result of an underflow. In such a case a corresponding interval computation would not deliver zero but a small interval with zero as one bound and a tiny positive or negative number as the other bound. In this case division is well defined by Table 1. The result is a closed interval which stretches continuously to $-\infty$ or $+\infty$ as the case may be.

From a theoretical point of view a note about the interval sets used in this paper and their relationships may be of interest. Basic to all considerations is the power set $\{P I\!\!R, \subseteq\}$. It is a complete lattice. The least element is the empty set $\varnothing$ and the greatest element is the set $I\!\!R = (-\infty, +\infty)$. The set $\{(I I\!\!R), \subseteq\}$ is an upper screen (a complete inf-subnet) of $\{P I\!\!R, \subseteq\}$. The set $\{(IF), \subseteq\}$ is a screen (a complete sublattice) of $\{(I I\!\!R), \subseteq\}$, see [4]. The sets $\{P I\!\!R, \subseteq\}$, $\{(I I\!\!R), \subseteq\}$, and $\{(IF), \subseteq\}$ have the same least and greatest element. With respect to set inclusion as an order relation upwardly directed roundings between the sets $P I\!\!R$ and $(I I\!\!R)$, and the sets $(I I\!\!R)$ and (IF) are used to define arithmetic operations in $(I I\!\!R)$ and (IF) via semimorphism. We briefly mention these properties here.

With the monotone upwardly directed roundings $\square : P I\!\!R \to (I I\!\!R)$ and $\Diamond : (I I\!\!R) \to (IF)$ arithmetic in $(I I\!\!R)$ and in (IF) is uniquely defined by the following properties :

(RG) $A \boxdot B := \square\,(A \circ B)$, for all $A, B \in (I I\!\!R)$ and all $\circ \in \{+, -, \cdot, /\}$.

(R1) $\square\, A = A$, for all $A \in (I I\!\!R)$,

(R2) $A \subseteq B \Rightarrow \square\, A \subseteq \square\, B$, for $A, B \in P I\!\!R$,

(R3) $A \subseteq \square\, A$, for all $A \in P I\!\!R$.

(RG) $A \mathbin{\Diamond\!\!\!\!\circ\;} B := \Diamond\,(A \boxdot B)$, for all $A, B \in (IF)$ and all $\circ \in \{+, -, \cdot, /\}$.

(R1) $\Diamond\, A = A$, for all $A \in (IF)$,

(R2) $A \subseteq B \Rightarrow \Diamond\, A \subseteq \Diamond\, B$, for $A, B \in (I I\!\!R)$,

(R3) $A \subseteq \Diamond\, A$, for all $A \in (I I\!\!R)$.

The resulting calculi are free of exceptions. The proof of this assertion is given by the formulas in the following section.

5 Complete Arithmetic for Intervals of (IF)

Addition	$(-\infty, b_2]$	$[b_1, b_2]$	$[b_1, +\infty)$	$(-\infty, +\infty)$
$(-\infty, a_2]$	$(-\infty, a_2 \overset{+}{\triangle} b_2]$	$(-\infty, a_2 \overset{+}{\triangle} b_2]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, a_2]$	$(-\infty, a_2 \overset{+}{\triangle} b_2]$	$[\boldsymbol{a_1} \overset{+}{\triangledown} \boldsymbol{b_1}, \boldsymbol{a_2} \overset{+}{\triangle} \boldsymbol{b_2}]$	$[a_1 \overset{+}{\triangledown} b_1, +\infty)$	$(-\infty, +\infty)$
$[a_1, +\infty)$	$(-\infty, +\infty)$	$[a_1 \overset{+}{\triangledown} b_1, +\infty)$	$[a_1 \overset{+}{\triangledown} b_1, +\infty)$	$(-\infty, +\infty)$
$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$

Table 5: Addition of extended intervals on the computer.

Subtraction	$(-\infty, b_2]$	$[b_1, b_2]$	$[b_1, +\infty)$	$(-\infty, +\infty)$
$(-\infty, a_2]$	$(-\infty, +\infty)$	$(-\infty, a_2 \overset{-}{\triangle} b_1]$	$(-\infty, a_2 \overset{-}{\triangle} b_1]$	$(-\infty, +\infty)$
$[a_1, a_2]$	$[a_1 \overset{-}{\triangledown} b_2, +\infty)$	$[\boldsymbol{a_1} \overset{-}{\triangledown} \boldsymbol{b_2}, \boldsymbol{a_2} \overset{-}{\triangle} \boldsymbol{b_1}]$	$(-\infty, a_2 \overset{-}{\triangle} b_1]$	$(-\infty, +\infty)$
$[a_1, +\infty)$	$[a_1 \overset{-}{\triangledown} b_2, +\infty)$	$[a_1 \overset{-}{\triangledown} b_2, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$

Table 6: Subtraction of extended intervals on the computer.

Multiplication	$[b_1, b_2]$ $b_2 \leq 0$	$[b_1, b_2]$ $b_1 < 0 < b_2$	$[b_1, b_2]$ $b_1 \geq 0$	$[0, 0]$	$(-\infty, b_2]$ $b_2 \leq 0$	$(-\infty, b_2]$ $b_2 \geq 0$	$[b_1, +\infty)$ $b_1 \leq 0$	$[b_1, +\infty)$ $b_1 \geq 0$	$(-\infty, +\infty)$
$[a_1, a_2], a_2 \leq 0$	$[\boldsymbol{a_2 \triangledown b_2, a_1 \vartriangle b_1}]$	$[\boldsymbol{a_1 \triangledown b_2, a_1 \vartriangle b_1}]$	$[\boldsymbol{a_1 \triangledown b_2, a_2 \vartriangle b_1}]$	$[0, 0]$	$[a_2 \triangledown b_2, +\infty)$	$[a_1 \triangledown b_2, +\infty)$	$(-\infty, a_1 \vartriangle b_1]$	$(-\infty, a_2 \vartriangle b_1]$	$(-\infty, +\infty)$
$a_1 < 0 < a_2$	$[\boldsymbol{a_2 \triangledown b_1, a_1 \vartriangle b_1}]$	$[\mathbf{min}(\boldsymbol{a_1 \triangledown b_2, a_2 \triangledown b_1}),$ $\mathbf{max}(\boldsymbol{a_1 \vartriangle b_1, a_2 \vartriangle b_2})]$	$[\boldsymbol{a_1 \triangledown b_2, a_2 \vartriangle b_2}]$	$[0, 0]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, a_2], a_1 \geq 0$	$[\boldsymbol{a_2 \triangledown b_1, a_1 \vartriangle b_2}]$	$[\boldsymbol{a_2 \triangledown b_1, a_2 \vartriangle b_2}]$	$[\boldsymbol{a_1 \triangledown b_1, a_2 \vartriangle b_2}]$	$[0, 0]$	$(-\infty, a_1 \vartriangle b_2]$	$(-\infty, a_2 \vartriangle b_2]$	$[a_2 \triangledown b_1, +\infty)$	$[a_1 \triangledown b_1, +\infty)$	$(-\infty, +\infty)$
$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
$(-\infty, a_2], a_2 \leq 0$	$[a_2 \triangledown b_2, +\infty)$	$(-\infty, +\infty)$	$(-\infty, a_2 \vartriangle b_1]$	$[0, 0]$	$[a_2 \triangledown b_2, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, a_2 \vartriangle b_1]$	$(-\infty, +\infty)$
$(-\infty, a_2], a_2 \geq 0$	$[a_2 \triangledown b_1, +\infty)$	$(-\infty, +\infty)$	$(-\infty, a_2 \vartriangle b_2]$	$[0, 0]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, +\infty), a_1 \leq 0$	$(-\infty, a_1 \vartriangle b_1]$	$(-\infty, +\infty)$	$[a_1 \triangledown b_2, +\infty)$	$[0, 0]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, +\infty), a_1 \geq 0$	$(-\infty, a_1 \vartriangle b_2]$	$(-\infty, +\infty)$	$[a_1 \triangledown b_1, +\infty)$	$[0, 0]$	$(-\infty, a_1 \vartriangle b_2]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$[a_1 \triangledown b_1, +\infty)$	$(-\infty, +\infty)$
$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$[0, 0]$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$

Table 7: Multiplication of extended intervals on the computer.

Division	$[b_1, b_2]$	$[b_1, b_2]$	$(-\infty, b_2]$	$[b_1, +\infty)$
$\boldsymbol{0 \notin B}$	$b_2 < 0$	$b_1 > 0$	$b_2 < 0$	$b_1 > 0$
$[a_1, a_2], a_2 \leq 0$	$[\boldsymbol{a_2} \bigtriangledown \boldsymbol{b_1}, \boldsymbol{a_1} \bigtriangleup \boldsymbol{b_2}]$	$[\boldsymbol{a_1} \bigtriangledown \boldsymbol{b_1}, \boldsymbol{a_2} \bigtriangleup \boldsymbol{b_2}]$	$[0, a_1 \bigtriangleup b_2]$	$[a_1 \bigtriangledown b_1, 0]$
$[a_1, a_2], a_1 < 0 < a_2$	$[\boldsymbol{a_2} \bigtriangledown \boldsymbol{b_2}, \boldsymbol{a_1} \bigtriangleup \boldsymbol{b_2}]$	$[\boldsymbol{a_1} \bigtriangledown \boldsymbol{b_1}, \boldsymbol{a_2} \bigtriangleup \boldsymbol{b_1}]$	$[a_2 \bigtriangledown b_2, a_1 \bigtriangleup b_2]$	$[a_1 \bigtriangledown b_1, a_2 \bigtriangleup b_1]$
$[a_1, a_2], a_1 \geq 0$	$[\boldsymbol{a_2} \bigtriangledown \boldsymbol{b_2}, \boldsymbol{a_1} \bigtriangleup \boldsymbol{b_1}]$	$[\boldsymbol{a_1} \bigtriangledown \boldsymbol{b_2}, \boldsymbol{a_2} \bigtriangleup \boldsymbol{b_1}]$	$[a_2 \bigtriangledown b_2, 0]$	$[0, a_2 \bigtriangleup b_1]$
$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
$(-\infty, a_2], a_2 \leq 0$	$[a_2 \bigtriangledown b_1, +\infty)$	$(-\infty, a_2 \bigtriangleup b_2]$	$[0, +\infty)$	$(-\infty, 0]$
$(-\infty, a_2], a_2 \geq 0$	$[a_2 \bigtriangledown b_2, +\infty)$	$(-\infty, a_2 \bigtriangleup b_1]$	$[a_2 \bigtriangledown b_2, +\infty)$	$(-\infty, a_2 \bigtriangleup b_1]$
$[a_1, +\infty), a_1 \leq 0$	$(-\infty, a_1 \bigtriangleup b_2]$	$[a_1 \bigtriangledown b_1, +\infty)$	$(-\infty, a_1 \bigtriangleup b_2]$	$[a_1 \bigtriangledown b_1, +\infty)$
$[a_1, +\infty), a_1 \geq 0$	$(-\infty, a_1 \bigtriangleup b_1]$	$[a_1 \bigtriangledown b_2, +\infty)$	$(-\infty, 0]$	$[0, +\infty)$
$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$

Table 8: Division of extended intervals with $0 \notin B$ on the computer.

Division	$B =$	$[b_1, b_2]$	$[b_1, b_2]$	$(-\infty, b_2]$	$[b_1, +\infty)$	
$\mathbf{0} \in \boldsymbol{B}$	$[0, 0]$	$b_1 < b_2 = 0$	$0 = b_1 < b_2$	$b_2 = 0$	$b_1 = 0$	$(-\infty, +\infty)$
$[a_1, a_2], a_2 < 0$	$\varnothing$	$[\boldsymbol{a_2} \bigtriangledown \boldsymbol{b_1}, +\infty)$	$(-\infty, \boldsymbol{a_2} \bigtriangleup \boldsymbol{b_2}]$	$[0, +\infty)$	$(-\infty, 0]$	$(-\infty, +\infty)$
$[a_1, a_2], a_1 \leq 0 \leq a_2$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, a_2], a_1 > 0$	$\varnothing$	$(-\infty, \boldsymbol{a_1} \bigtriangleup \boldsymbol{b_1}]$	$[\boldsymbol{a_1} \bigtriangledown \boldsymbol{b_2}, +\infty)$	$(-\infty, 0]$	$[0, +\infty)$	$(-\infty, +\infty)$
$(-\infty, a_2], a_2 < 0$	$\varnothing$	$[a_2 \bigtriangledown b_1, +\infty)$	$(-\infty, a_2 \bigtriangleup b_2]$	$[0, +\infty)$	$(-\infty, 0]$	$(-\infty, +\infty)$
$(-\infty, a_2], a_2 > 0$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, +\infty), a_1 < 0$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, +\infty), a_1 > 0$	$\varnothing$	$(-\infty, a_1 \bigtriangleup b_1]$	$[a_1 \bigtriangledown b_2, +\infty)$	$(-\infty, 0]$	$[0, +\infty)$	$(-\infty, +\infty)$
$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$

Table 9: Division of extended intervals with $0 \in B$ on the computer.

The rules for the operations of extended intervals on the computer in Tables 5—9 look rather complicated. Their implementation seems to require many case distinctions. The situation, however, can be greatly simplified as follows.

On the computer actually only the basic rules for addition, subtraction, multiplication, and division for closed and bounded intervals of IF including division by an interval that includes zero need to be provided. In Tables 5—9 these rules are printed in bold letters.

The remaining rules shown in the tables can automatically be produced out of these basic rules by the computer itself if a few well established rules for computing with $-\infty$ and $+\infty$ are formally applied. With $x \in F$ these rules are

$$\begin{array}{ll}
\infty + x = \infty, & -\infty + x = -\infty, \\
-\infty + (-\infty) = (-\infty) \cdot \infty = -\infty, & \infty + \infty = \infty \cdot \infty = \infty, \\
\infty \cdot x = \infty \text{ for } x > 0, & \infty \cdot x = -\infty \text{ for } x < 0, \\
\frac{x}{\infty} = \frac{x}{-\infty} = 0, &
\end{array}$$

together with variants obtained by applying the sign rules and the law of commutativity.

If in an interval operand a bound is $-\infty$ or $+\infty$ the multiplication with 0 is performed as if the following rules would hold

$$0 \cdot (-\infty) = 0 \cdot (+\infty) = (-\infty) \cdot 0 = (+\infty) \cdot 0 = 0. \tag{5.1}$$

These rules have no meaning otherwise.

We stress that (5.1) does not define new rules for the multiplication of 0 with $+\infty$ or $-\infty$. It just describes a short cut for applying the continuity principle mentioned earlier in this section.

6 An Alternative Approach

In two cases (rows 4 and 7) in Table 1 the result consists of the union of two distinct sets. In these two cases zero is an interior point of the divisor. In Section 3 we recommended splitting the interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ into two distinct sets $[b_1, 0)$ and $(0, b_2]$ and performing the division sequentially. The result then consists of the two distinct sets $(-\infty, c_2]$ and $[c_1, +\infty)$.

This is not the usual way the division is performed in the literature. In the user's program division by an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ appears as one single operation. The division, therefore, is done at once. With existing processors only one interval can be delivered as the result of an interval operation. On the computer, therefore, the result is returned as an improper interval $[c_1, c_2]$ where the left hand bound is higher than the right hand bound $c_1 > c_2$.

This raises the question of whether arithmetic operations for improper intervals should be defined. This is not recommended. The situation can occur repeatedly. Zero can again be an interior point of one of the distinct intervals. The result of division would then consist of the union of three distinct sets and so on.

In the literature an improper interval $[c_1, c_2]$ with $c_1 > c_2$ occasionally is called an 'exterior interval'. On the number circle an 'exterior interval' is interpreted as an interval with infinity as an interior point. We do not follow this line here. Interval arithmetic is defined as an arithmetic for sets of real numbers. Operations for real numbers which deliver ∞ as their result do not exist. Here and in the following the symbols $-\infty$ and $+\infty$ are only used to describe sets of real numbers.

Independently of whether division by an interval $[b_1, b_2]$ with $b_1 < 0 < b_2$ is performed at once or sequentially in two steps the result consists of two distinct sets. Also, in our interpretation an improper interval describes the two distinct sets $(-\infty, c_2]$ and $[c_1, +\infty)$. In any case the flag that the result consists of two *distinct intervals* should be raised and signaled to the user. The user may then take measures to deal with the situation as is appropriate for his application.

7 Realization of Complete Interval Arithmetic on the Computer

For completeness we summarize in this section the formulas or rules that should be used to realize Complete Interval Arithmetic on the computer. These are the rules that should be required by an interval arithmetic standard.

Basic for all interval computations on the computer are the rules for addition, subtraction, multiplication and division including division by an interval that contains zero for closed and bounded intervals of IF:

Addition $[a_1, a_2] + [b_1, b_2] = [a_1 \mathbin{\bigtriangledown\!\!\!\!\!\!+} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\!+} b_2]$.

Subtraction $[a_1, a_2] - [b_1, b_2] = [a_1 \mathbin{\bigtriangledown\!\!\!\!\!\!-} b_2, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\!-} b_1]$.

Multiplication	$[b_1, b_2]$ $b_2 \le 0$	$[b_1, b_2]$ $b_1 < 0 < b_2$	$[b_1, b_2]$ $b_1 \ge 0$
$[a_1, a_2], a_2 \le 0$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_2, a_1 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_1]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_2, a_1 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_1]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_2, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_1]$
$a_1 < 0 < a_2$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_1, a_1 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_1]$	$[min(a_1 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_2, a_2 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_1),$ $max(a_1 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_2)]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_2, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_2]$
$[a_1, a_2], a_1 \ge 0$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_1, a_1 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_2]$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_2]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!\cdot} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!\cdot} b_2]$

Division $\mathbf{0} \notin \boldsymbol{B}$	$[b_1, b_2]$ $b_2 < 0$	$[b_1, b_2]$ $b_1 > 0$
$[a_1, a_2], a_2 \le 0$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!/} b_1, a_1 \mathbin{\bigtriangleup\!\!\!\!\!/} b_2]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!/} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!/} b_2]$
$[a_1, a_2], a_1 < 0 < a_2$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!/} b_2, a_1 \mathbin{\bigtriangleup\!\!\!\!\!/} b_2]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!/} b_1, a_2 \mathbin{\bigtriangleup\!\!\!\!\!/} b_1]$
$[a_1, a_2], a_1 \ge 0$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!/} b_2, a_1 \mathbin{\bigtriangleup\!\!\!\!\!/} b_1]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!/} b_2, a_2 \mathbin{\bigtriangleup\!\!\!\!\!/} b_1]$

Division $\mathbf{0} \in \boldsymbol{B}$	$B =$ $[0, 0]$	$[b_1, b_2]$ $b_1 < b_2 = 0$	$[b_1, b_2]$ $0 = b_1 < b_2$
$[a_1, a_2], a_2 < 0$	$\varnothing$	$[a_2 \mathbin{\bigtriangledown\!\!\!\!\!/} b_1, +\infty)$	$(-\infty, a_2 \mathbin{\bigtriangleup\!\!\!\!\!/} b_2]$
$[a_1, a_2], a_1 \le 0 \le a_2$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$[a_1, a_2], a_1 > 0$	$\varnothing$	$(-\infty, a_1 \mathbin{\bigtriangleup\!\!\!\!\!/} b_1]$	$[a_1 \mathbin{\bigtriangledown\!\!\!\!\!/} b_2, +\infty)$

A design for a hardware unit that realizes these operations is given in [3]. With it interval arithmetic would be about as fast as simple floating-point arithmetic.

Division by an interval that includes zero leads to extended intervals. Arithmetic operations for extended intervals of (IF) can be performed on the computer if in addition a few formal rules for operations with $+\infty$ and $-\infty$ are applied in the computer. These rules are shown in the following tables.

Any operation with the empty set has the empty set as its result.

Addition	$-\infty$	b	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-$
a	$-\infty$	$-$	$+\infty$
$+\infty$	$-$	$+\infty$	$+\infty$

Subtraction	$-\infty$	b	$+\infty$
$-\infty$	$-$	$-\infty$	$-\infty$
a	$+\infty$	$-$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	$-$

Multiplication	$-\infty$	$b < 0$	0	$b > 0$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	0	$-\infty$	$-\infty$
$a < 0$	$+\infty$	$-$	$-$	$-$	$-\infty$
0	0	$-$	$-$	$-$	0
$a > 0$	$-\infty$	$-$	$-$	$-$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	0	$+\infty$	$+\infty$

Division	$-\infty$	$+\infty$
a	0	0

8 Comparison Relations and Lattice Operations

Three comparison relations are important for intervals of IF and (IF):

$$\textit{equality}, \quad \textit{less than or equal}, \quad \text{and} \quad \textit{set inclusion}. \tag{8.1}$$

Let A and B be intervals of (IF) with bounds $a_1 \leq a_2$ and $b_1 \leq b_2$ respectively. Then the relations *equality* and *less than or equal* in (IF) are defined by:

$$\begin{aligned} A = B \quad &:\Leftrightarrow a_1 = b_1 \wedge a_2 = b_2, \\ A \leq B \quad &:\Leftrightarrow a_1 \leq b_1 \wedge a_2 \leq b_2. \end{aligned}$$

Since bounds for intervals of (IF) may be $-\infty$ or $+\infty$ these comparison relations are executed as if performed in the lattice $\{F^*, \leq\}$ with $F^* := F \cup \{-\infty\} \cup \{+\infty\}$.

With the order relation $\leq$, $\{(IF), \leq\}$ is a lattice. The *greatest lower bound* (glb) and the *least upper bound* (lub) of $A, B \in (IF)$ are the intervals

$$\begin{aligned} glb(A, B) \quad &:= [min(a_1, b_1), min(a_2, b_2)], \\ lub(A, B) \quad &:= [max(a_1, b_1), max(a_2, b_2)]. \end{aligned}$$

The greatest lower bound and the least upper bound of an interval with the empty set are both the empty set.

The inclusion relation in (IF) is defined by

$$A \subseteq B :\Leftrightarrow b_1 \leq a_1 \wedge a_2 \leq b_2. \tag{8.2}$$

With the relation $\subseteq$, $\{(IF), \subseteq\}$ is also a lattice. The least element in $\{(IF), \subseteq\}$ is the empty set $\emptyset$ and the greatest element is the interval $(-\infty, +\infty)$. The infimum of two elements $A, B \in (IF)$ is the intersection and the supremum is the interval hull (convex hull):

$$\begin{aligned} inf(A, B) \quad &:= [max(a_1, b_1), min(a_2, b_2)] \quad \text{or the empty set } \emptyset, \\ sup(A, B) \quad &:= [min(a_1, b_1), max(a_2, b_2)]. \end{aligned}$$

The intersection of an interval with the empty set is the empty set. The interval hull with the empty set is the other operand.

If in the formulas for $glb(A, B)$, $lub(A, B)$, $inf(A, B)$, $sup(A, B)$, a bound is $-\infty$ or $+\infty$ a parenthesis should be used at this interval bound to denote the resulting interval. This bound is not an element of the interval.

If in any of the comparison relations defined here both operands are the empty set, the result is true. If in (8.2) A is the empty set the result is true. Otherwise the result is false if in any of the three comparison relations only one operand is the empty set.[6]

A particular case of inclusion is the relation *element of.* It is defined by

$$a \in B :\Leftrightarrow b_1 \leq a \wedge a \leq b_2. \tag{8.3}$$

Another useful check is for whether an interval $[a_1, a_2]$ is a proper interval, that is, if $a_1 \leq a_2$.

9 Evaluation of Functions

Interval evaluation of real functions fits smoothly into complete interval arithmetic as developed in the previous sections. Let f be a function and D_f its domain of definition. For an interval $X \subseteq D_f$, the range $f(X)$ of f is defined as the set of the function's values for all $x \in X$:

$$f(X) := \{f(x) | x \in X\}. \tag{9.1}$$

A function $f(x) = a/(x - b)$ with $D_f = I\!R \setminus \{b\}$ is sometimes called singular or discontinuous at $x = b$. Both descriptions are meaningless in a strict mathematical sense. Since $x = b$ is not of the domain of f, the function cannot have any property at $x = b$.

In this strict sense a division $2/[b_1, b_2]$ by an interval $[b_1, b_2]$ that contains zero as an interior point, $b_1 < 0 < b_2$, means:
$2/([b_1, 0) \cup (0, b_2]) = 2/[b_1, 0) \cup 2/(0, b_2] = (-\infty, 2/b_1] \cup [2/b_2, +\infty)$.

We give two examples:

$f(x) = 4/(x - 2)^2, \quad D_f = I\!R \setminus \{2\}, \quad X = [1, 4],$
$f([1, 2) \cup (2, 4]) = f([1, 2)) \cup f((2, 4]) = [4, +\infty) \cup [1, +\infty) = [1, +\infty).$

$g(x) = 2/(x - 2), \quad D_g = I\!R \setminus \{2\}, \quad X = [1, 3],$
$g([1, 2) \cup (2, 3]) = g([1, 2)) \cup g((2, 3]) = (-\infty, -2] \cup [2, +\infty),$
Here the flag *distinct intervals* should be raised and signaled to the user. The user may then choose a routine to apply which is appropriate for the application.

It has been suggested in the literature that the entire set of real numbers $(-\infty, +\infty)$ be returned as result in this case. However, this may be a large overestimation of the true result and there are applications (Newton's method) which need the accurate answer. To return the entire set of real numbers is also against a basic principle of interval arithmetic—to keep the sets as small as possible. So a standard should have the most accurate answer returned.

On the computer, interval evaluation of a real function $f(x)$ for $X \subseteq D_f$ should deliver a highly accurate enclosure of the range $f(X)$ of the function.

Evaluation of a function $f(x)$ for an interval X with $X \cap D_f = \varnothing$, of course, does not make sense, since $f(x)$ is not defined for values outside its domain D_f.

[6]A convenient encoding of the empty set may be $\varnothing = [+NaN, -NaN]$. Then most comparison relations and lattice operations considered in this section would deliver the correct answer if conventional rules for NaN are applied. However, if $A = \varnothing$ then set inclusion (8.2) and computing the interval hull do not follow this rule. So in these two cases whether $A = \varnothing$ must be checked before the operations can be executed. However, to avoid confusion with the IEEE P754 exceptions we recommend to use another notation $\varnothing = [\varnothing_1, \varnothing_2]$ with similar properties.

The empty set $\varnothing$ should be delivered and an error message may be given to the user.

There are, however, applications in interval arithmetic where information about a function f is useful when X exceeds the domain D_f of f. The interval X may also be the result of overestimation during an earlier interval computation.

In such cases the range of f can only be computed for the intersection $X' := X \cap D_f$:

$$f(X') := f(X \cap D_f) := \{f(x) | x \in X \cap D_f\}. \tag{9.2}$$

To prevent the wrong conclusions being drawn, the user must be informed that the interval X had to be reduced to $X' := X \cap D_f$ to compute the delivered range. A particular flag for *domain overflow* may serve this purpose. An appropriate routine can be chosen and applied if this flag is raised.
We give a few examples:

$l(x) := log(x), \quad D_{log} = (0, +\infty)$,
$log((0, 2]) = (-\infty, log(2)]$.
But also
$log([-5, 2]') = log((0, 2]) = (-\infty, log(2)]$.
The flag *domain overflow* should be set. It informs the user that the function has been evaluated for the intersection $X' := X \cap D_f = [-5, 2] \cap (0, +\infty) = (0, 2]$.

$h(x) := sqrt(x), \quad D_{sqrt} = [0, +\infty)$,
$sqrt([1, 4]) = [1, 2]$,
$sqrt([4, +\infty)) = [2, +\infty)$.
$sqrt([-5, -1]) = \varnothing$, an error message $sqrt$ not defined for $[-5, -1]$, may be given to the user.
$sqrt([-5, 4]') = sqrt([0, 4]) = [0, 2]$.
The flag *domain overflow* should be set. It informs the user that the function has been evaluated for the intersection $X' := X \cap D_f = [-5, 4] \cap [0, +\infty) = [0, 4]$.

$k(x) := sqrt(x) - 1, \quad D_k = [0, +\infty)$,
$k([-4, 1]') = k([0, 1]) = sqrt([0, 1]) - 1 = [-1, 0]$.
The flag *domain overflow* should be set. It informs the user that the function has been evaluated for the intersection $X' := X \cap D_f = [-4, 1] \cap [0, +\infty) = [0, 1]$.

10 Final Remarks

The basis of all technical computing is the set of real numbers $\mathbb{R}$. In general, however, real numbers are not representable and operations for them are not executable. On the computer the real numbers are mapped on a finite subset F, floating-point numbers. Floating-point arithmetic and interval arithmetic are different tools that approximate arithmetic for real numbers.

Floating-point arithmetic has been used very successfully. A lot of mathematics is used in conventional Numerical Analysis. Nevertheless the result of a long floating-point computation always carries some uncertainty. In contrast to this interval mathematics even on the computer is supposed to deliver results with guarantees. This is only possible if it is strictly kept on mathematical grounds.

It has been shown in this paper that arithmetic operations and comparison relations for closed intervals of $(I\mathbb{R})$ and (IF) can be derived in a strict mathematical manner. Arithmetic operations of $(I\mathbb{R})$ are defined as power set operations. Formulas for the operations for extended intervals of $(I\mathbb{R})$ are obtained by a continuity principle via the operations for closed and bounded intervals of $I\mathbb{R}$. Formulas for the operations of (IF) are obtained from those of $(I\mathbb{R})$ by rounding the result in $(I\mathbb{R})$ on the least upper bound in (IF) with respect to set inclusion as an order relation. The tables in Section 5 show that the result of any operation for operands of (IF) leads to an element of (IF) again, i.e., arithmetic in (IF) is free of exceptions. In Section 7 it is shown that only a small set of operations suffices to realize all arithmetic operations for operands of (IF) on the computer.

Interval arithmetic solely deals with closed and connected sets of real numbers. $-\infty$ and $+\infty$ do not occur as elements of intervals. They are only used as bounds of intervals. This is essential for the entire calculus to be free of exceptions.

While in the real number field division by zero is not defined, in IEEE floating-point arithmetic division by zero is defined to be $+\infty$ or $-\infty$. In interval arithmetic there is no need for this definition. Here division of a number $a \neq 0$ by zero is well defined. The result is the empty set.

The tremendous progress in computer technology and the great increase in computer speed should be accompanied by extension of the mathematical capacity of the computer. Today floating-point arithmetic is hardware supported on every powerful processor. This should also hold for interval arithmetic. With very very little additional arithmetic circuitry interval arithmetic can be made available at the speed of simple floating-point arithmetic. See [3] and the letter of the IFIP Working Group 2.5 to IEEE754R [5]. Hardware support for the interval operations summarized in Section 7 would greatly simplify a future interval arithmetic standard.

Floating-point numbers and floating-point intervals are objects of contrasting quality. A floating-point number is an approximate representation of a real number, while an interval is a precisely defined object. An operation mixing the two, which ought to yield an interval, may not be precisely specified. It is thus not reasonable to define operations between floating-point numbers and intervals. The two calculi should be kept strictly separate. If a user does indeed need to perform an operation between a floating-point number and a floating-point interval, he may do so by using a transfer function which transforms its floating-point operand into a floating-point interval. In doing so, the user is made aware of the possible loss of quality of the interval as a precise object. The transfer function should check whether a given pair of floating-point values is a correct interval. Prohibiting an automatic type transfer from floating-point numbers to floating-point intervals also prevents exceptions of the IEEE floating-point arithmetic standard from being introduced into interval arithmetic. A strict separation of the two calculi would not slow down the runtime of interval computations if the operations summarized in Sections 7 are realized in hardware.

Of course, computing with result verification often makes use of floating-point computations.

Interval arithmetic for a fixed precision should be accompanied by a variable length interval arithmetic. For definition see [4]. The basic tool to achieve this is an exact multiply and accumulate (i.e., continued addition) operation for the data format double precision. Pipelining gives it high speed and exactitude brings very high accuracy into computation. There is no way to compute a dot product faster than the exact result. By pipelining it can be computed in the time the processor needs to read the data, i.e., it could not be done faster. Variable length interval arithmetic fully benefits from such speed. No software simulation can go as fast. With operator overloading variable length interval arithmetic is very easy to use. The exponent range can be kept very flexible, see [4] and the literature listed there.

References

[1] Alefeld, G., Herzberger, J.: *Introduction to Interval Computations.* Academic Press, New York, 1983.

[2] Kahan, W.: *A More Complete Interval Arithmetic.* Lecture Notes prepared for a summer course at the University of Michigan, June 17-21, 1968.

[3] Kirchner, R., Kulisch, U.: *Hardware support for interval arithmetic.* Reliable Computing 12:3, 225–237, 2006.

[4] Kulisch, U.,W.: *Computer Arithmetic and Validity – Theory, Implementation and Applications.* De Gruyter, Berlin, New York, 2008.

[5] IFIPWG-IEEE754R: *Letter of the IFIP WG 2.5 to the IEEE Computer Arithmetic Revision Group, 2007.*[7]

[6] Moore, R. E.: *Interval Analysis.* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1966.

[7] Moore, R. E.: *Methods and Applications of Interval Analysis.* SIAM, Philadelphia, Pennsylvania, 1979.

[8] Ratz, D.: *On Extended Interval Arithmetic and Inclusion Isotony.* Preprint, Institut für Angewandte Mathematik, Universität Karlsruhe, 1999.

[9] Rump, S.M.: *Kleine Fehlerschranken bei Matrixproblemen.* Dissertation, Universität Karlsruhe, 1980.

[7] See the author's homepage, http://www.mathematik.uni-karlsruhe.de/ianm2/∼kulisch.