## 4. Level 1 description

In this clause, subclauses 4.1 to 4.5 describe the theory of mathematical intervals and interval functions that underlies this standard. The relation between expressions and the point or interval functions that they define is specified, since it is is central to the Fundamental Theorem of Interval Arithmetic. Subclauses 4.6, 4.7 list the required and recommended *arithmetic operations* (also called elementary functions) with their mathematical specifications. Subclause 4.8 describes, at a mathematical level, the system of *decorations* that is used for exception handling in P1788.

4.1. **Numbers.** Following the terminology of 754 (754§2.1.25 and elsewhere), any member of the extended reals $\mathbb{R} \cup \{-\infty, +\infty\}$ is called a number: it is a **finite number** if it is in the reals $\mathbb{R}$, else an **infinite number**.

4.2. **Intervals.** The set of mathematical intervals in $\mathbb{R}$, denoted $\overline{\mathbb{IR}}$, comprises[1] all closed intervals of real numbers

$$\boldsymbol{x} = [\underline{x}, \overline{x}] := \{\, x \in \mathbb{R} \mid \underline{x} \le x \le \overline{x} \,\},$$

where the bounds $\underline{x}, \overline{x}$ are extended-real numbers, $-\infty \le \underline{x} \le \overline{x} \le +\infty$.
[*Notes.*

- *In particular, the empty interval* $[-\infty, -\infty] = [+\infty, +\infty] = \emptyset$ *belongs to* $\overline{\mathbb{IR}}$.
- *The above definition implies* $-\infty$ *and* $+\infty$ *can be bounds of an interval, but are never members of it.*
- *The round bracket or outward bracket notations for closed intervals in* $\mathbb{R}$ *with an infinite end point (e.g.,* $[2, +\infty)$ *or* $[2, +\infty[$ *instead of* $[2, +\infty]$*) are not used in this document but are not considered wrong.*

]

4.3. **Hull.** The (interval) **hull** of an arbitrary subset $\boldsymbol{s}$ of $\mathbb{R}$, written hull($\boldsymbol{s}$), is the tightest member of $\overline{\mathbb{IR}}$ that contains $\boldsymbol{s}$. (The **tightest** set with a given property is the intersection of all sets having that property, provided the intersection itself has this property.)

4.4. **Expressions and functions.**

4.4.1. *Function terminology.* In this standard, operations are written as named functions; in a specific implementation they might be represented by operators (i.e., using some form of infix notation), or by families of format-specific functions, or by operators or functions whose names might differ from those in this standard.

The terms operation, function and mapping are broadly synonymous; however the usage in this standard is as follows. The next subclauses make precise definitions of these terms.

- A (point) *arithmetic operation* is a mathematical real function for which an implementation provides versions in a collection of user-available operations called the implementation's *library*. This includes functions normally written in operator form (e.g., +, ×) and those normally written in function form (e.g., exp, arctan). It is not specified how a particular implementation provides library facilities.
- An *interval arithmetic operation* is an interval version of a point arithmetic operation, provided by a implementation in its library. There are also *decorated interval arithmetic operations*, introduced in 4.8.
- A *constructor* is a function that creates an interval from non-interval data.
- An *interval non-arithmetic operation* is a mapping of intervals to intervals that is provided by an implementation but is not an interval version of a point function (e.g., the operation of intersection of two intervals).
- The term *function* includes the arithmetic operations, and more generally functions constructed from these by expressions in a user program. It is also used for non-interval-valued operations on an interval, such as the function that returns the midpoint of an interval.

[1] The overline is for compatibility with older notation, which uses $\mathbb{IR}$ for the set of closed and *bounded* real intervals.

4.4.2. *Expressions.* An **expression** in zero or more (independent) *variables* is a symbolic object defined to be either one of those variables or, recursively, of the form $\phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$ where $\phi$ represents an *operation* taking $k$ arguments ($k \geq 0$), and the $\mathsf{g}_i$ are expressions. Other syntax may be used, such as traditional algebraic notation and/or program pseudo-code. An **arithmetic expression** is one in which all the operations are arithmetic operations. An expression may be denoted by a sans-serif letter, e.g. $\mathsf{f}$, to distinguish it from a corresponding function $f$ that it defines.

The set $\operatorname{vars}\mathsf{f}$ of variables that **occur** in $\mathsf{f}$ is defined recursively as follows where the variable-names are regarded as distinct, atomic, symbols. If $\mathsf{f}$ is a variable $z$ then $\operatorname{vars}\mathsf{f}$ is the singleton $\{z\}$. Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$ then $\operatorname{vars}\mathsf{f} = \operatorname{vars}(\mathsf{g}_1) \cup \ldots \cup \operatorname{vars}(\mathsf{g}_k)$. If the variables are an indexed set, e.g. $z_1, \ldots, z_n$, one may use a set of indices $i$ instead of the corresponding set of $z_i$. [*Example.* $\operatorname{vars}\big((e^x + e^{-x})/(2y)\big) = \{x, y\}$, *while* $\operatorname{vars}\big((e^{z_1} + e^{-z_1})/(2z_3)\big)$ *can be written as* $\{z_1, z_3\}$ *or as* $\{1, 3\}$, *whichever is convenient.*]
[*Example.*

– *The object* $\mathsf{f}$ *where*

$$\mathsf{f} = (e^x + e^{-x})/(2y)$$

*is an arithmetic expression in two variables* $x, y$ *that may be written in the above form as*

$$\operatorname{div}(\operatorname{plus}(\exp(\mathsf{x}), \exp(\operatorname{uminus}(\mathsf{x}))), \operatorname{times}(2(), \mathsf{y})),$$

*where* uminus, plus, times, div *and* exp *name the arithmetic operations "unary minus", +, ×, ÷ and exponential function. The literal constant 2 is regarded as a zero-argument arithmetic operation* $2()$, *see 4.4.6.*

*The expression may be broken (e.g., by a compiler) into elementary operations that may be sequenced in several ways, one of which is*

$$\begin{aligned} v_1 &= \exp(x) \\ v_2 &= \operatorname{uminus}(x) \\ v_3 &= \exp(v_2) \\ v_4 &= \operatorname{plus}(v_1, v_3) \\ v_5 &= 2() \\ v_6 &= \operatorname{times}(v_5, y) \\ f &= \operatorname{div}(v_4, v_6). \end{aligned}$$

*All these forms are regarded as defining the same* $\mathsf{f}$.

– *An expression that uses the interval intersection or union operation is not an arithmetic expression.*

]

An optional *formal argument list* may be specified by notation such as

$$\mathsf{f}(\mathsf{z}_1, \ldots, \mathsf{z}_n) = \text{an expression},.$$

This defines $\mathsf{f}$ to be the indicated expression with the formal argument list $\mathsf{z}_1, \ldots, \mathsf{z}_n$, which must include at least the variables in $\operatorname{vars}\mathsf{f}$, but possibly others.
[*Example. A formal argument list is useful when making* $\mathsf{f}$ *define a function. The specifications*

$$\begin{aligned} \mathsf{f}(x, y) &= (e^x + e^{-x})/(2y), \\ \mathsf{f}(y, x) &= (e^x + e^{-x})/(2y), \\ \mathsf{f}(w, x, y, z) &= (e^x + e^{-x})/(2y) \end{aligned}$$

*are all valid and all different (they would define different functions), while*

$$\mathsf{f}(y) = (e^x + e^{-x})/(2y)$$

*is incorrect since the argument list does not include* $\operatorname{vars}\mathsf{f}$ *(unless* $x$ *is separately defined to be a parameter, not a variable).* ]

4.4.3. *Expressions and program code.* To define an arithmetic expression, a section of program code must only use arithmetic operations. If also it consists of straight-line code, then it certainly defines an arithmetic expression. Certain kinds of loops and branches are permitted, but a function defined by implicit code, e.g. by a root-finding iteration, is generally excluded. See Annex C for a fuller discussion.

4.4.4. *Point functions.* A **point function** is a (possibly partial) multivariate real function: that is, a mapping $f$ from a subset $D$ of $\mathbb{R}^n$ to $\mathbb{R}^m$ for some integers $n \geq 0, m > 0$. When not otherwise specified, a scalar function is assumed, i.e. $m = 1$. If $m > 1$, the function is called a vector function. The set $D$ where $f$ is defined is its **domain**, also written $\operatorname{dom} f$ . To specify $n$, call $f$ an $n$-variable point function, or denote values of $f$ as

$$f(x_1, \ldots, x_n). \tag{1}$$

The **range** of $f$ over an arbitrary subset $\boldsymbol{s}$ of $\mathbb{R}^n$ is the set

$$\operatorname{range}(f, \boldsymbol{s}) = \{\, f(x) \mid x \in \boldsymbol{s} \text{ and } x \in \operatorname{dom} f \,\}. \tag{2}$$

Thus mathematically, when evaluating a function over a set, points outside the domain are ignored (e.g., $\operatorname{range}(\text{sqrt}, [-1, 1]) = [0, 1]$ ).

Equivalently, for the case where $f$ takes separate arguments $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$, each being a subset of $\mathbb{R}$, the range is written as $\operatorname{range}(f, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$. This is an alternative notation when $\boldsymbol{s}$ is the cartesian product of the $\boldsymbol{s}_i$.

4.4.5. *Interval mappings and functions.* A box is an interval vector $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \overline{\mathbb{IR}}^n$. It is usually identified with the cartesian product $\boldsymbol{x}_1 \times \ldots \times \boldsymbol{x}_n \subseteq \mathbb{R}^n$; however, the correspondence is only one-to-one when all the $\boldsymbol{x}_j$ are nonempty.

Given an $n$-variable point function $f$, an **interval extension** of $f$, also called an **interval version** of $f$, is a mapping $\boldsymbol{f}$ from boxes $\boldsymbol{x} \in \overline{\mathbb{IR}}^n$ to intervals, such that

$$\boldsymbol{f}(\boldsymbol{x}) \supseteq \operatorname{range}(f, \boldsymbol{x})$$

for any such box $\boldsymbol{x}$, regarded as a subset of $\mathbb{R}^n$. The **sharp interval extension** of $f$ is defined by

$$\boldsymbol{f}(\boldsymbol{x}) = \operatorname{hull}(\operatorname{range}(f, \boldsymbol{x})).$$

Equivalently, using multiple-argument notation for $f$, an interval extension satisfies

$$\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \supseteq \operatorname{range}(f, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n),$$

and the sharp interval extension is defined by

$$\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \operatorname{hull}(\operatorname{range}(f, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n))$$

for any intervals $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

In some contexts it is useful for $\boldsymbol{x}$ to be a general subset of $\mathbb{R}^n$, or the $\boldsymbol{x}_i$ to be general subsets of $\mathbb{R}$; the definition is unchanged.

Note that such a function is automatically defined for all (interval or set) arguments. The decoration system of 4.8 gives a systematic way to diagnose when the underlying point function has been evaluated outside its domain.

When $f$ is a binary operator $\bullet$ written in infix notation, this gives the usual definition of its sharp interval extension as

$$\boldsymbol{x} \bullet \boldsymbol{y} = \operatorname{hull}(\{\, x \bullet y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y}, \text{ and } x \bullet y \text{ is defined} \,\}).$$

[*Example. With these definitions, the relevant sharp interval extensions satisfy* $\sqrt{[-1, 4]} = [0, 2]$ *and* $\sqrt{[-2, -1]} = \emptyset$*; also* $\boldsymbol{x} \times \{0\} = \{0\}$ *for any nonempty* $\boldsymbol{x}$*, and* $\boldsymbol{x}/\{0\} = \emptyset$*, for any* $\boldsymbol{x}$*.*]

When $f$ is a vector point function, a vector interval function with the same number of inputs and outputs as $f$ is called an interval extension of $f$ if each of its components is an interval extension of the corresponding component of $f$.

A mapping with interval inputs and outputs is called an **interval function** if it is an interval version of some point function, and an interval mapping otherwise. An **interval arithmetic operation** is an interval version of a point arithmetic operation. An interval mapping, provided by an implementation but not an interval arithmetic operation, is called an **interval non-arithmetic operation** (e.g., interval intersection and union, $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x} \cap \boldsymbol{y}$ and $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \operatorname{hull}(\boldsymbol{x} \cup \boldsymbol{y})$ ).

*Decorated* interval functions and arithmetic and non-arithmetic operations, see 4.8, are defined similarly.

4.4.6. *Constants.* A real scalar function with no arguments—a mapping $\mathbb{R}^n \to \mathbb{R}^m$ with $n = 0$ and $m = 1$—is a **real constant**. Languages may distinguish between a literal constant (e.g. the decimal value defined by the string `1.23e4`) and a named constant (e.g. $\pi$) but the difference is not relevant in this standard.

Taking this view of constants has two advantages. First, it automatically defines the notion of an interval version of a constant. (E.g., any interval containing $\pi$ is an interval extension of $\pi$; the sharp extension is $[\pi, \pi]$.)

Second, this formalism gives a Level 1 definition of NaN, "Not a Number". $\mathbb{R}^0$ is the zero-dimensional vector space $\{0\}$ (i.e., it has one element, conventionally named 0). The real numbers $r$ are in one-to-one correspondence with the mappings $0 \mapsto r$, so that $\mathbb{R}$ can be identified with those real constants that are *total* functions $\mathbb{R}^0 \to \mathbb{R}$. However there is one *non-total* such function, namely the one with empty domain and, therefore, no value. This empty function is called NaN.

It follows from the definitions that the sharp *interval extension* of NaN is the empty interval $\emptyset$. It is shown in C.2 that its sharp *decorated interval extension* has properties appropriate to a "Not an Interval" object, and is therefore called NaI.

## 4.5. Functions defined by expressions.

4.5.1. *Generic functions.* A single arithmetic operation name such as +, or an arithmetic expression such as $\mathsf{f}(x, y) = x + \sin y$, is used to refer to different, related, functions: such operations and expressions, or the resulting functions, are called generic (or polymorphic). Whether generic function syntax can be used in program code is language-defined.

In this standard, a given arithmetic operation (see 4.6, 4.7) or function defined by an arithmetic expression exists in any of the following versions:

(a) A point function. This is unique, theoretical and generally non-computable.
(b) An interval function. These are computable and generally non-unique: e.g., there is at least one for each supported finite precision interval type.
(c) A decorated interval function, similarly to the previous.

The operation or expression may also denote a floating point function in any supported format, but this is not relevant here.

Importantly for applying the Fundamental Theorems below, the arguments of a function defined by an expression $\mathsf{f}$ come from the variables that *actually occur* in $\mathsf{f}$. See Annex C for details. [*Example. User code may define a generic function $f(x, y)$ specifying a surface $z = f(x, y)$ in 3D. If the actual expression for $f$ involves $x$ only, then as far as the theorems are concerned the result is a one-variable function of $x$ only.*]



4.5.2. *Point, interval and decorated interval functions.* It is assumed that the method of associating actual to formal arguments for the $n$ variables occurring in an arithmetic expression $\mathsf{f}$ is positional, by listing them $z_1, \ldots, z_n$ in some standard order. Languages may use other ways (e.g. keyword association).

The **point function of** (or defined by) $\mathsf{f}$ is a function $f$ from $\mathbb{R}^n$ to $\mathbb{R}$. Its value at an actual argument $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is defined as follows:

– If $\mathsf{f}$ is the variable $\mathsf{z}_i$, the value is the number $x_i$.
– Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$, the value is the point version of $\phi$ evaluated at $(u_1, \ldots, u_n)$ where $u_i$ is the value of the point function of $\mathsf{g}_i$ at $(x_1, \ldots, x_n)$.

An **interval function of** $\mathsf{f}(\mathsf{z}_1, \ldots, \mathsf{z}_n)$ is a function from $\overline{\mathbb{IR}}^n$ to $\overline{\mathbb{IR}}$, again called $f$. Its value at an actual argument, a box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \overline{\mathbb{IR}}^n$, is defined as follows:

– If $\mathsf{f}$ is the variable $\mathsf{z}_i$, the value is the interval $\boldsymbol{x}_i$.
– Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$, the value is some interval version of $\phi$ evaluated at $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ where $\boldsymbol{u}_i$ is the value of the interval function of $\mathsf{g}_i$ at $\boldsymbol{x}$.

A **decorated interval function of** $\mathsf{f}(\mathsf{z}_1, \ldots, \mathsf{z}_n)$ is as the previous, with "decorated interval" replacing "interval", see 4.8.

4.5.3. *Disambiguation.* The phrase "some interval version" in the definition of an interval function of an expression $\mathsf{f}$ is used, because there are many such versions. The **sharp** or **exact arithmetic** interval function of $\mathsf{f}$ is the unique (generally non-computable) one in which the sharp interval extension of each arithmetic operation $\phi$ is used at each evaluation step. Otherwise, a rule must be

specified for choosing a particular version of each $\phi$. In a program context, this rule is language-defined. The same generic $\phi$ may be evaluated by different interval versions at different points of an expression.

The same considerations apply to decorated interval functions of f.

[*Example. Consider evaluation of* $(x + y) + z$ *at interval actual arguments* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$*, and suppose* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are of a single precision interval type while* $\boldsymbol{z}$ *is of a double precision type. Depending on language rules, an implementation might first convert* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *to double precision so that both occurrences of "+" use the same, double precision, interval function. Or it might do the first addition in single, then convert the result to double and do the second addition in double, so that the generic "+" translates to two different interval functions.* ]

4.5.4. *Domain.* The **domain** of a generic function is defined as follows. For a point arithmetic operation $\phi$, its domain $\operatorname{dom}\phi$ is part of its definition (4.6, 4.7). For the point function $f$ of an arithmetic expression, the **natural domain** $\operatorname{dom} f$ is defined to be the set of points where the expression makes sense, that is, at which it can be evaluated without going outside the domain of any of its constituent operations.

[*Example. The natural domain of* $f(x, y) = 1/(\sqrt{x-1} - y)$ *is the set of* $(x, y)$ *in the plane that do not cause either square root of a negative number or division by zero, i.e. where* $x \geq 1$ *and* $y \neq \sqrt{x-1}$.]

For an interval or decorated interval function of an arithmetic expression, the question of domain does not arise: such functions are automatically defined for all input arguments.

## 4.8. **The decoration system.**

4.8.1. *The purpose of decorated intervals.* Decorations are properties of a function for which an interval enclosure of its range over a box is being computed. There are two main objectives of interval calculations:

– obtaining correct range enclosures for a real-valued function $f$ of real variables;
– verifying the assumptions of existence, uniqueness, or nonexistence theorems.

While traditional interval analysis targets the first aim, decorations target the second aim.

From an interval evaluation of a function $f$ at an interval $\boldsymbol{x}$, one not only wants to get a range enclosure $f(\boldsymbol{x})$ but also a guarantee that the pair $(f, \boldsymbol{x})$ has certain important properties, such as $f(\boldsymbol{x})$ being well-formed, $f(x)$ being defined for all $x \in \boldsymbol{x}$, $f$ restricted to $\boldsymbol{x}$ being continuous or bounded, etc. This goal is achieved, in parts of a program that require it, by adding to each interval extra information called a **decoration**, whose semantics is summarized as follows:

> Each intermediate step of the original computation depends on some or all of the inputs, so it can be viewed as an intermediate function of these inputs. The result interval obtained on each intermediate step is an enclosure for the range of the corresponding intermediate function. The decoration attached to this intermediate interval reflects the available knowledge about whether this intermediate function is guaranteed to be everywhere defined, continuous, bounded, etc., on the given inputs. For example, we may place the decoration "defined" only if a rigorous argument ensures that the intermediate function is indeed defined in the box determined by the input intervals.

The P1788 decoration model, in contrast with 754's, has no global flags. A general aim, as in 754's use of NaN and flags, is not to interrupt the flow of computation: rather, to collate information during evaluation, that may be inspected after it.

In particular, this enables a fully local handling of exceptional conditions in interval calculations, an important feature in a concurrent computing environment.

Note that a decoration primarily describes a property, not of the interval it is attached to, but of the function defined by a section of code that produced that interval.

The system is outlined here at a mathematical level, with the finite-precision aspects in 5.3 and a fuller discussion of the theory in the informative Annex C. Subclause 4.8.2 gives the basic definitions and 4.8.3 describes the mathematics that underpins the decoration model. Subclause 4.9 is on the algebraic aspect of the P1788 decoration model: decorated intervals and (bare) intervals and decorations are unified in a single abstract datatype that supports operations on any mix of these. This has implications for implementation and language support, discussed in Annex C.

In the context of decorations, "model" means the concepts in 4.8.2 to 4.8.5 and the unified algebraic structure in 4.9; "system" means all this plus the resulting user-available features, and their conceptual and implemented infrastructure.

4.8.2. *Definitions.* The set $\mathbb{D}$ of **decorations** has five elements:

| Value | Short description |
|---|---|
| `saf` | safe |
| `def` | defined |
| `con` | containing |
| `emp` | empty |
| `ill` | ill-formed |

Formally, each $d \in \mathbb{D}$ represents the set of pairs $(f, \boldsymbol{x})$ consisting of a real-valued function $f$ with domain $\operatorname{dom} f \subseteq \mathbb{R}^n$ for some $n$ and a box $\boldsymbol{x} \in \overline{\mathbb{IR}}^n$ for which the property $p_d(f, \boldsymbol{x})$ is valid. Here

$$\begin{aligned}
&p_{\texttt{saf}}(f, \boldsymbol{x}) : \boldsymbol{x} \text{ is a nonempty subset of } \operatorname{dom} f, \text{ and the restriction of } f \\
&\qquad\qquad \text{to } \boldsymbol{x} \text{ is continuous and bounded;} \\
&p_{\texttt{def}}(f, \boldsymbol{x}) : \boldsymbol{x} \text{ is a nonempty subset of } \operatorname{dom} f; \\
&p_{\texttt{con}}(f, \boldsymbol{x}) : \text{always true;} \\
&p_{\texttt{emp}}(f, \boldsymbol{x}) : \boldsymbol{x} \text{ is disjoint from } \operatorname{dom} f \\
&\qquad\qquad (\text{either or both of } \boldsymbol{x} \text{ and } \operatorname{dom} f \text{ may be empty}); \\
&p_{\texttt{ill}}(f, \boldsymbol{x}) : \operatorname{dom} f \text{ is empty.}
\end{aligned} \tag{4}$$

The decorations are partially ordered by *strength*, where stronger means smaller, regarding the decorations in the above way as sets:

$$\texttt{con} \supseteq \texttt{def} \supseteq \texttt{saf}, \quad \texttt{con} \supseteq \texttt{emp} \supseteq \texttt{ill}. \tag{5}$$

Thus `con` is the weakest decoration, where nothing is claimed; `saf` and `ill` claim the most and are strongest.

Decorations are also given a total ordering according to *quality*:

$$\texttt{ill} < \texttt{emp} < \texttt{con} < \texttt{def} < \texttt{saf}, \tag{6}$$

with `ill` the "worst" and `saf` the "best". When taking the minimum or maximum of decorations, this ordering is implied.

A **decorated interval** is a pair, written interchangeably as $(\boldsymbol{x}, d)$ or $\boldsymbol{x}_d$, where $\boldsymbol{x} \in \overline{\mathbb{IR}}$ is a real interval and $d \in \mathbb{D}$ is a decoration, such that the following *Invariant Property* IP holds:

$$\text{IP:} \qquad d \text{ is in } \{\,\texttt{saf}, \texttt{def}, \texttt{con}\,\} \text{ if } \boldsymbol{x} \text{ is nonempty, and in } \{\,\texttt{emp}, \texttt{ill}\,\} \text{ if } \boldsymbol{x} \text{ is empty.} \tag{7}$$

The set of decorated intervals is denoted $\overline{\mathbb{DIR}}$. An interval or decoration may be referred to as a *bare* interval or decoration, to emphasize that it is not a decorated interval.

$\boldsymbol{x}_d$ may also be a decorated interval vector, where $\boldsymbol{x}$ and $d$ are vectors of intervals $\boldsymbol{x}_i$ and decorations $d_i$ respectively, defining decorated interval components $(\boldsymbol{x}_i, d_i)$ of $\boldsymbol{x}_d$. The set of decorated interval vectors of length $n$ is denoted $\overline{\mathbb{DIR}}^n$.

In program format, e.g. pseudocode, for a decorated interval named `x`, the interval part is written `x.box` and the decoration part `x.dec`.

A **containment order** $\supseteq$ is defined componentwise:

$$(\boldsymbol{x}, d) \supseteq (\boldsymbol{x}', d') \quad \text{iff} \quad \boldsymbol{x} \supseteq \boldsymbol{x}' \text{ and } d \supseteq d', \tag{8}$$

using set inclusion for the first component and the order (5) for the second component.

As used in the Fundamental Theorem below, $\boldsymbol{x}'$ need not be a real interval but can be a general subset of $\mathbb{R}$. In this case $(\boldsymbol{x}', d')$ is termed a **decorated set**.

4.8.3. *The Fundamental Theorems.* The original theorem due to Ramon Moore (see 1.3) may be stated in the terminology of 4.5 as follows:

**Theorem 4.1** (Fundamental Theorem of Interval Arithmetic, FTIA)**.** *Let* `f` *be an arithmetic expression. If the interval function of each arithmetic operation in* `f` *is an interval extension of its corresponding point function, then any interval function of* `f` *is an interval extension of the point function of* `f`.



The decoration system provides an extended theorem in terms of the concept of a *decorated interval extension* of a function, which is now defined.

Let $f$ be a function and $\boldsymbol{x}$ a (bare) box as in 4.8.2. The **decoration of $f$ over $\boldsymbol{x}$**, written $\mathrm{dec}(f, \boldsymbol{x})$, is defined to be the strongest decoration $d$ for which $p_d(f, \boldsymbol{x})$ is true. That is,

$$\mathrm{dec}(f, \boldsymbol{x}) = \begin{cases} \texttt{saf} & \text{if } p_{\texttt{saf}}(f, \boldsymbol{x}) \text{ holds;} \\ \texttt{def} & \text{if } p_{\texttt{saf}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{def}}(f, \boldsymbol{x}) \text{ holds;} \\ \texttt{ill} & \text{if } p_{\texttt{ill}}(f, \boldsymbol{x}) \text{ holds;} \\ \texttt{emp} & \text{if } p_{\texttt{ill}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{emp}}(f, \boldsymbol{x}) \text{ holds;} \\ \texttt{con} & \text{otherwise.} \end{cases} \tag{9}$$

This is well-defined because $(f, \boldsymbol{x})$ cannot qualify for both a "good" decoration (`saf` or `def`) and a "bad" one (`emp` or `ill`).

Note that though `con` is "trivial", to have $\mathrm{dec}(f, \boldsymbol{x}) = \texttt{con}$ is not trivial: it asserts $p_{\texttt{emp}}$ and $p_{\texttt{def}}$ are both false. In particular $\boldsymbol{x}$ is not a single point, as it meets both $\mathrm{dom}\, f$ and its complement.

The **decoration $\mathrm{dec}(f, \boldsymbol{x}_c)$ of $f$ over a decorated box $\boldsymbol{x}_c$** is defined by the formula:

$$\mathrm{dec}(f, \boldsymbol{x}_c) = \min\{c_0, c_1, \ldots, c_n\}, \quad \text{where } c_0 = \mathrm{dec}(f, \boldsymbol{x}), \tag{10}$$

and the minimum is with respect to the quality order (6). That is, the decoration of a function over a *decorated* box equals the worst of its decoration over the *bare* box and the decorations of all the box components. ⚠ An awkward point. The formula (10) looks arbitrary and depends on the particular set $\mathbb{D}$ of decorations chosen. I have tried to produce an "intrinsic" definition that leads to (10) for our particular $\mathbb{D}$ and would adapt to other $\mathbb{D}$s. See discussion in C.1.3.

For a point function $f$ from $\mathbb{R}^n$ to $\mathbb{R}$ and a decorated box $\boldsymbol{x}_c \in \overline{\mathbb{DIR}}^n$ as above, the **decorated range** of $f$ over $\boldsymbol{x}_c$, written $\mathrm{drange}(f, \boldsymbol{x}_c)$, is the pair

$$\mathrm{drange}(f, \boldsymbol{x}_c) = (\mathrm{range}(f, \boldsymbol{x}),\ \mathrm{dec}(f, \boldsymbol{x}_c)). \tag{11}$$

It is a decorated set, as defined earlier, consisting of the range of $f$ over the bare box, and the decoration of $f$ over the decorated box. A **decorated interval extension**, or **decorated interval version**, of $f$ is a mapping $\boldsymbol{f}$ from decorated boxes $\boldsymbol{x}_d \in \overline{\mathbb{DIR}}^n$ to decorated intervals, such that

$$\boldsymbol{f}(\boldsymbol{x}_d) \supseteq \mathrm{drange}(f, \boldsymbol{x}_d)$$

for any such decorated box $\boldsymbol{x}_d$, where its interval part is regarded as a subset of $\mathbb{R}^n$, and containment is for the interval and decoration components separately, as defined in (8).

The generalized Moore's theorem is as follows. It is proved in Annex C.

**Theorem 4.2** (Fundamental Theorem of Decorated Interval Arithmetic, FTDIA)**.** *Let* `f` *be an arithmetic expression. If the decorated interval function of each arithmetic operation in* `f` *is a decorated interval extension of its corresponding point function, then any decorated interval function of* `f` *is a decorated interval extension of the point function of* `f`*.*

Just as the FTIA gives a computable enclosure of the usually non-computable range of a function over a box, so the FTDIA gives (along with enclosing the range) a computable enclosure, in the sense of (5), of the usually non-computable properties of a function being everywhere defined, continuous, etc., over a box.

To obtain maximum information about $f$ over a particular bare box, each of its components should be initialized with the "best"—in the quality order—decoration consistent with its value. This is done by the **domain**() function which converts a bare to a decorated interval as follows:

$$\mathbf{domain}(\boldsymbol{x}) = \boldsymbol{x}_d \text{ where } d = \begin{cases} \texttt{saf} & \text{if } \boldsymbol{x} \text{ is a nonempty bounded interval,} \\ \texttt{def} & \text{if } \boldsymbol{x} \text{ is nonempty unbounded interval,} \\ \texttt{emp} & \text{if } \boldsymbol{x} \text{ is empty.} \end{cases} \tag{12}$$

For a box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, this function acts componentwise, producing a decorated box $\boldsymbol{x}_d$ where $(\boldsymbol{x}_d)_i = \mathbf{domain}(\boldsymbol{x}_i)$, $i = 1, \ldots, n$.

If the $\boldsymbol{x}_i$ are given any other permitted decoration, the conclusions listed below are still valid, but may be more cautious than necessary. Thus normal application of the FTDIA, given an expression $f(x_1, \ldots, x_n)$ and a bare box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, consists of the following steps:

1. Form $\boldsymbol{x}_d = \mathbf{domain}(\boldsymbol{x})$;
2. Form $\boldsymbol{y}_e = f(\boldsymbol{x}_d)$ by decorated interval evaluation;
3. Inspect the decoration $e$.

The conclusions one can draw are as follows, where $f$ denotes the point function of the expression, $Y = \mathrm{range}(f, \boldsymbol{x})$ is its true range over $\boldsymbol{x}$, and $D = \mathrm{dec}(f, \boldsymbol{x})$ is its corresponding true decoration:

- If $d =$ `saf`, then $D$ must also be `saf`, so $f$ is guaranteed everywhere defined, continuous and bounded on $\boldsymbol{x}$, which is nonempty.
- If $d =$ `def`, then $D$ must be `def` or `saf`: $f$ is guaranteed everywhere defined on $\boldsymbol{x}$, which is nonempty. It might be continuous and/or bounded there as well, but this is not known.
- If $d =$ `ill`, then $D$ must be `ill`: $\mathrm{dom}\, f$ is empty, that is, $f$ is nowhere defined. This can only occur if some component of $\boldsymbol{x}_c$ was constructed, not from a bare interval using **domain**(), but from something ill-formed. This has a special significance of *Not an Interval* at computational level: see C.2.
- If $d =$ `emp`, then $D$ must be `emp` or `ill`: $\mathrm{dom}\, f$ is guaranteed to be disjoint from $\boldsymbol{x}$. This includes the cases where $\boldsymbol{x}$ is empty, which has no special name, and where $\mathrm{dom}\, f$ is empty, which is the `ill` case.
- If $d =$ `con`, then $D$ might be any of the five values: no conclusion can be drawn.

When $f$ is vector-valued, then $\boldsymbol{y}_d$ is a decorated interval vector, and the above applies componentwise.

[*Example. Consider decorated interval evaluation of* $f(x, y) = \sqrt{x \times (y - x) - 1}$ *with various input intervals* $\boldsymbol{x}$, $\boldsymbol{y}$, *using the strict decorated interval extension of each arithmetic operation. The natural domain* $\mathrm{dom}\, f$ *is easily seen to the union of the regions* $x > 0$, $y \geq x + \dfrac{1}{x}$ *and* $x < 0$, $y \leq x + \dfrac{1}{x}$.

*For manageable notation, agree that an interval named* $\boldsymbol{x}$ *is given a decoration* $dx$, *and so on.*

*(i) Let* $\boldsymbol{x} = [1,2]$, $\boldsymbol{y} = [3,4]$, *defining a box* $(\boldsymbol{x},\boldsymbol{y})$ *contained in* dom $f$. *Applying the* **domain** *function gives initial decorated intervals* $\boldsymbol{x}_{dx} = [1,2]_{\texttt{saf}}$, $\boldsymbol{y}_{dy} = [3,4]_{\texttt{saf}}$. *The first operation is*

$$\boldsymbol{u}_{du} = \boldsymbol{y}_{dy} - \boldsymbol{x}_{dx} \qquad = [1,3]_{\texttt{saf}}.$$

*Namely, subtraction is defined and continuous on all of* $\mathbb{R}^2$, *and bounded on bounded rectangles (call this property "nice" for short), so the bare result decoration is* $du' = \mathrm{dec}(-,(\boldsymbol{y},\boldsymbol{x})) = \texttt{saf}$, *whence by (10) the (strict, i.e. best possible) decoration on* $\boldsymbol{u}$ *is* $du = \min\{du', dy, dx\} = \min\{\texttt{saf},\texttt{saf},\texttt{saf}\} = \texttt{saf}$. *Multiplication is also "nice", so the second operation similarly gives*

$$\boldsymbol{v}_{dv} = \boldsymbol{x}_{dx} \times \boldsymbol{u}_{du} \qquad = [1,6]_{\texttt{saf}}.$$

*The constant 1, following 4.4.6, becomes a decorated interval function returning the constant value* $[1,1]_{\texttt{saf}}$. *The next operation is again "nice", and gives*

$$\boldsymbol{w}_{dw} = \boldsymbol{v}_{dv} - 1 \qquad = [0,5]_{\texttt{saf}}$$

*Finally* $\sqrt{\cdot}$ *is defined, continuous and bounded on* $\boldsymbol{w} = [0,5]$, *so, arguing similarly, one has the final result*

$$\boldsymbol{f}_{df} = \sqrt{\boldsymbol{w}_{dw}} \qquad = [0,\sqrt{5}]_{\texttt{saf}}.$$

*That was the* **mechanism**. *According to the FTDIA it provides a rigorous* **proof** *that for the box* $\boldsymbol{z}_{dz} = (\boldsymbol{x}_{dx},\boldsymbol{y}_{dy}) = ([1,2]_{\texttt{saf}},[3,4]_{\texttt{saf}})$,

$\boldsymbol{u}_{du} = [1,3]_{\texttt{saf}}$ *encloses the decorated range of* $u(x,y) = y - x$ *over* $\boldsymbol{z}_{dz}$;
$\boldsymbol{v}_{dv} = [1,6]_{\texttt{saf}}$ *encloses the decorated range of* $v(x,y) = x(y-x)$ *over* $\boldsymbol{z}_{dz}$;
$\boldsymbol{w}_{dw} = [0,5]_{\texttt{saf}}$ *encloses the decorated range of* $w(x,y) = x(y-x)-1$ *over* $\boldsymbol{z}_{dz}$;
*and finally*
$\boldsymbol{f}_{df} = [0,\sqrt{5}]_{\texttt{saf}}$ *encloses the decorated range of* $f(x,y) = \sqrt{x(y-x)-1}$ *over* $\boldsymbol{z}_{dz}$.

*The final result says*

$$[0,\sqrt{5}] \supseteq \mathsf{range}(f,\boldsymbol{z}),$$
$$\texttt{saf} \supseteq \mathsf{dec}(f,\boldsymbol{z}_{dz}) = \min(\mathsf{dec}(f,\boldsymbol{z}),\texttt{saf},\texttt{saf}) = \mathsf{dec}(f,\boldsymbol{z}),$$

*whence,* $\mathsf{dec}(f,\boldsymbol{z}) = \texttt{saf}$. *That is,* $f(x,y)$ *is defined, continuous and bounded on the box* $1 \le x \le 2,\ 3 \le y \le 4$, *and its range over this box is enclosed in* $[0,\sqrt{5}]$.

*(ii) Let* $\boldsymbol{x} = [1,2]$ *as before, but* $\boldsymbol{y} = [\frac{5}{2},4]$. *The box* $\boldsymbol{z}$ *is still contained in* dom $f$ *so the true value of* $\mathsf{dec}(f,\boldsymbol{z})$ *is still* saf. *However the evaluation fails to detect this because of interval widening due to "dependency". Namely after* $\boldsymbol{u}_{du} = [\frac{5}{2},3]_{\texttt{saf}}$, $\boldsymbol{v}_{dv} = [\frac{5}{2},6]_{\texttt{saf}}$, $\boldsymbol{w}_{dw} = [-\frac{1}{2},5]_{\texttt{saf}}$, *the final result has interval part* $\boldsymbol{f} = \sqrt{[-\frac{1}{2},5]} = [0,\sqrt{5}]$ *as before, but* $\sqrt{\cdot}$ *is not everywhere defined on* $\boldsymbol{w}$, *so that* $dw' = \mathrm{dec}(\sqrt{\cdot},\boldsymbol{w}) = \mathrm{dec}(\sqrt{\cdot},[-\frac{1}{2},5]) = \texttt{con}$ *giving* $\mathrm{dec}(\sqrt{\cdot},\boldsymbol{w}_{dw}) = \min\{dw',dv\} = \texttt{con}$, *so finally* $\boldsymbol{f}_{df} = [0,\sqrt{5}]_{\texttt{con}}$. *This is a valid enclosure of the decorated range* $[0,\sqrt{5}]_{\texttt{saf}}$, *but less useful in this case.*

*(iii) If* $\boldsymbol{x} = [1,2]$, $\boldsymbol{y} = [1,1]$, *the box* $\boldsymbol{z}$ *is now wholly outside* dom $f$, *and evaluation detects this, giving the exact result* $\boldsymbol{f}_{df} = \emptyset_{\texttt{emp}}$. *However, if* $\boldsymbol{x} = [1,2]$, $\boldsymbol{y} = [1,\frac{3}{2}]$, *the box is still wholly outside* dom $f$, *but owing to widening, evaluation fails to detect this, giving* $\boldsymbol{f}_{df} = [0,0]_{\texttt{con}}$. *This is still a valid enclosure of the decorated range* $\emptyset_{\texttt{emp}}$, *but "too wide" to be of any use.*

]

4.8.4. *User-supplied functions.* A user program may define a decorated interval version of a point function, to be used within expressions as if it were a library function. This does not invalidate the FTDIA, subject to the one requirement that the decorated interval version be a decorated interval extension of the point function.

[*Example. In some applications, an interval extension of the function defined by*

$$\psi(x) = x + 1/x$$

*is required. The expression as it stands can give poor enclosures: e.g., with* $\boldsymbol{x} = [\frac{1}{2},2]$, *one obtains*

$$\psi(\boldsymbol{x}) = [\tfrac{1}{2},2] + 1/[\tfrac{1}{2},2] = [\tfrac{1}{2},2] + [\tfrac{1}{2},2] = [1,4],$$

*which is much wider than* $\mathsf{range}(\psi,\boldsymbol{x}) = [2,2\frac{1}{2}]$.

*Thus it is useful to code a tight enclosure by special methods, e.g. monotonicity arguments, and*

*provide this as a new library function. Suppose this has been done. To implement a decorated interval extension just entails adding code to compute the decoration $d$ of the point function $x + 1/x$ over an input decorated interval $\boldsymbol{x}_c$, on the lines of the following:*

*// compute decoration $c_0$ over bare interval $\boldsymbol{x}$:*
1. *if $\boldsymbol{x}$ is empty or singleton $[0, 0]$ then $c_0 =$* `emp`*;*
2. *elseif $0 \in \boldsymbol{x}$ then $c_0 =$* `con`*;*
3. *elseif $\boldsymbol{x}$ is unbounded then $c_0 =$* `def`*;*
4. *else $c_0 =$* `saf`*;*

*// combine with input decoration by equation (10):*
5. $d = \min\{c_0, c\}$.

]

4.8.5. *Non-arithmetic operations.* The propagation of decorations under non-arithmetic operations is unspecified by the definitions of the semantics of decorations in 4.8.2. This must therefore be analyzed on different grounds, including important examples of use in algorithms. For the intersection and union operations, an analysis is given in Annex C. Its conclusions are as follows.

The specification for **intersection** is intended for the situation where a function (or an intermediate term in a function) can be expressed by two algebraically equivalent expressions, neither of which always gives a tighter enclosure than the other in interval mode. Then one may evaluate both forms and intersect the results. This leads to

$$\boldsymbol{g}_{dg} \cap \boldsymbol{h}_{dh} = \boldsymbol{f}_{df}$$

where
- if $dg =$ `ill` then $\boldsymbol{f}_{df} = \boldsymbol{h}_{dh}$,
- if $dh =$ `ill` then $\boldsymbol{f}_{df} = \boldsymbol{g}_{dg}$,
- elseif $df =$ `emp` or $dg =$ `emp` or $\boldsymbol{g} \cap \boldsymbol{h} = \emptyset$ then $\boldsymbol{f}_{df} = (\emptyset, \texttt{emp})$,
- else $\boldsymbol{f}_{df} = (\boldsymbol{g} \cap \boldsymbol{h}, \max(dg, dh))$.

The specification for **union** (i.e., the interval hull) is intended for the situation where a function (or an intermediate term in a function) is expressed for different ranges of an argument by different expressions. This leads to

$$\boldsymbol{g}_{dg} \cup \boldsymbol{h}_{dh} = \boldsymbol{f}_{df}$$

where
- if $dg =$ `ill` or $dh =$ `ill` then $\boldsymbol{f}_{df} = (\emptyset, \texttt{emp})$,
- elseif $df = dg =$ `emp` then $\boldsymbol{f}_{df} = (\emptyset, \texttt{emp})$,
- else $\boldsymbol{f}_{df} = (\boldsymbol{g} \cup \boldsymbol{h}, \max(\texttt{con}, \min(dg, dh)))$.

4.9. **Unified interval-decoration opoerations.** The notion of interval extension gives a way to convert an elementary operation with real arguments and result, say $z = x \bullet y$, to one with bare interval arguments and result, $\boldsymbol{z} = \boldsymbol{x} \bullet \boldsymbol{y}$, and one with decorated interval arguments and result, $\boldsymbol{z}_d = \boldsymbol{x}_b \bullet \boldsymbol{y}_c$. The P1788 model goes further. Each interval operation is "unified" to act on an arbitrary mix of bare intervals, bare decorations and decorated intervals, and give one of these as result.

An important motive for this is practical. There are significant applications where one wishes to compute an interval result, provided certain exceptions don't occur. Say, one wishes to find an interval $\boldsymbol{y}$ that encloses the range of a function $f$ over a box $\boldsymbol{x}$, but only if $f$ is everywhere defined on the box. During evaluation of an arithmetic expression, as soon as a value worse than `def` occurs one knows that $p_{\texttt{def}}(f, \boldsymbol{x})$ is surely false. It is then permissible to discard the interval value, and propagate the decoration through the evaluation instead, as a diagnostic.

Probably the most used interval type will have a 16 byte bare interval and a decorated interval 17 byte decorated interval. The latter is likely to be slow to access and wasteful to store. One can mitigate this by letting the efficient 16-byte object store *either* a bare interval *or* a bare decoration—which is easily done at Level 3—though not both at once. It then becomes essential to support operations on a mix of the two.

⚠ Still to be revised in the light of work by Hayes and Neumaier, so I have omitted most of it.

## Appendix C. Decoration system: details and examples

### C.1. The Fundamental Theorem of Decorated Interval Arithmetic.

#### C.1.1. *Statement and notation.*

The FTDIA was stated in 4.8.3 as follows.

**Theorem C.1** (Fundamental Theorem of Decorated Interval Arithmetic, FTDIA)**.** *Let* f *be an arithmetic expression. If the decorated interval function of each arithmetic operation in* f *is a decorated interval extension of its corresponding point function, then any decorated interval function of* f *is a decorated interval extension of the point function of* f*.*

The proof involves several lemmas.

First, it is necessary to resolve a technical difficulty in the relation between expressions and functions, due to the empty set being a valid interval in this standard.

[*Example. Consider the vector function* $f = (f_1, f_2)$ *where* $f(x, y) = (e^x, x + y)$*, evaluated on the box* $B = (\boldsymbol{x}, \boldsymbol{y}) = ([0, 1], \emptyset)$*. The standard interpretation of the FTIA is that the result of interval evaluation,*

$$f(\boldsymbol{x}, \boldsymbol{y}) = (e^{[0,1]}, [0, 1] + \emptyset) = ([1, e], \emptyset),$$

*encloses, componentwise, the ranges of the point functions* $f_1$ *and* $f_2$ *over the box* $B$*, regarding the latter as a subset of* $\mathbb{R}^2$*, namely the cartesian product* $\boldsymbol{x} \times \boldsymbol{y}$*. In this example,* $B$ *is empty, hence so are both ranges, and in this form the FTIA gives the useless conclusion that* $[1, e] \supseteq \emptyset$ *and* $\emptyset \supseteq \emptyset$*.*

*Introducing decorations makes things worse: the first enclosure becomes* $[1, e]_{\texttt{saf}} \supseteq \emptyset_{\texttt{emp}}$*, which is actually false because* `saf` *does not enclose* `emp`*.*

*The problem is due to treating both components of* $f$ *as functions of both* $x$ *and* $y$*, when in fact* $f_1$ *depends only on* $x$*.*]

Using *sparse vector* notation removes this problem. Let the variables used be named $z_1, \ldots, z_n$ and (see 4.4.2) represent the set vars f of variables that occur in an expression f by the corresponding integer indices, a subset $I$ of $\{1, ..., n\}$. If $X$ is some set, a sparse $X$-vector (indexed by $I$) is a member $x$ of $X^I$, the space of $X$-valued mappings on $I$. E.g., a sparse real vector $x$ indexed by $I = \{2, 5, 7\}$ has three real-valued elements denoted $x_2$, $x_5$ and $x_7$; other elements are "absent", or regarded as filled with NaNs.

A point, interval or decorated interval function of an expression f will take arguments that are sparse $X$-vectors indexed over vars f, where $X$ is $\mathbb{R}$, $\overline{\mathbb{IR}}$ or $\overline{\mathbb{DIR}}$, respectively, and the definitions of expression evaluation in 4.5.2 are revised as follows.

The **point function of** f is a function $f$ from $\mathbb{R}^{\text{vars f}}$ to $\mathbb{R}$. Let $x \in \mathbb{R}^{\text{vars f}}$. If f is a variable $z_i$ (so vars f $= \{i\}$), then $f(x)$ is the number $x_i$. Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$ (so vars f is the union of vars $\mathsf{g}_1, \ldots,$ vars $\mathsf{g}_k$) then $f(x)$ is the value of the point version of $\phi$ at $(u_1, \ldots, u_k)$ where $u_i$ is the value of the point function of $\mathsf{g}_i$ at $x[\text{vars}(\mathsf{g}_i)]$. The latter notation means the sub-vector of $x$ comprising those $x_i$ for which $i \in \text{vars}(\mathsf{g}_i)$—formally, the restriction of map $x$ to vars$(\mathsf{g}_i)$.

An **interval function of** f is any function $f$ from $\overline{\mathbb{IR}}^{\text{vars f}}$ to $\overline{\mathbb{IR}}$ that obeys the following. Let $\boldsymbol{x} \in \overline{\mathbb{IR}}^{\text{vars f}}$. If f is a variable $z_i$, then $f(\boldsymbol{x})$ is any interval enclosing $\boldsymbol{x}_i$. Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$ then $f(\boldsymbol{x})$ is the value of some interval version of $\phi$ at $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ where $\boldsymbol{u}_i$ is the value of some interval function of $\mathsf{g}_i$ at $\boldsymbol{x}[\text{vars}(\mathsf{g}_i)]$.

A **decorated interval function of** f is any function $f$ from $\overline{\mathbb{DIR}}^{\text{vars f}}$ to $\overline{\mathbb{DIR}}$ that obeys the following. Let $\boldsymbol{x}_b \in \overline{\mathbb{DIR}}^{\text{vars f}}$. If f is a variable $z_i$, then $f(\boldsymbol{x}_b)$ equals $(\boldsymbol{x}_b)_i$ (the *strict* extension), or in general any interval enclosing $(\boldsymbol{x}_b)_i$ in the decorated interval sense.

⚠ This is not quite true on my present definition of decorated interval extension—specifically of dec$(f, \boldsymbol{x}_b)$. It forces $\boldsymbol{x}_{\texttt{saf}}$ to become $\boldsymbol{x}_{\texttt{def}}$ if $\boldsymbol{x}$ is unbounded, which is unsatisfactory!

Recursively, if $\mathsf{f} = \phi(\mathsf{g}_1, \ldots, \mathsf{g}_k)$ then $f(\boldsymbol{x}_b)$ is the value of some decorated interval version of $\phi$ at $\boldsymbol{u}_c = ((\boldsymbol{u}_c)_1, \ldots, (\boldsymbol{u}_c)_k)$ where $(\boldsymbol{u}_c)_i = (\boldsymbol{u}_i, c_i)$ is the value of some decorated interval function of $\mathsf{g}_i$ at $\boldsymbol{x}_b[\text{vars}(\mathsf{g}_i)]$.

[*Note. Issues of subscripting aside, the base case of these definitions—where* f *is a single variable—has the identity* $id(u) = u$*, for real* $u$*, as its point function, so the requirement on the base case for the interval versions is that the [decorated] interval function of the identity be a [decorated] interval extension of the point* $id()$ *function. This can be achieved by regarding* $id()$ *as a member of the arithmetic operation library.* ]

C.1.2. *Proof.*

⚠ Note on the Lemmas. I haven't tried to put Lemma C.2 into "sparse vector" notation because I'm not convinced that is the best way to go. So at present there is a disconnect between Lemma C.2 and Lemma C.3. But I think Lemma C.3 is consistent with the main proof that follows.

**Lemma C.2.** *Let $g : \mathbb{R}^n \to \mathbb{R}^k$ and $\phi : \mathbb{R}^k \to \mathbb{R}$ be multivariate real functions and define the function $f : \mathbb{R}^n \to \mathbb{R}$ by $f = \phi \circ g$, that is*

$$f(x) = \phi(g(x)), \tag{15}$$

*where $x = (x_1, \ldots, x_n)$ and $g(x) = (g_1(x), \ldots, g_k(x))$. The functions may be partial, so each $g_i$ is defined on its domain $\operatorname{dom} g_i \subseteq \mathbb{R}^n$, and $\phi$ on its domain $\operatorname{dom} \phi \subseteq \mathbb{R}^k$.*

*Let $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \overline{\mathbb{IR}}^n$ be a box. Let $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \in \overline{\mathbb{IR}}^k$ be a box whose components satisfy*

$$\boldsymbol{u}_i \supseteq \operatorname{range}(g_i, \boldsymbol{x}) \quad \text{for } i = 1, \ldots, k. \tag{16}$$

*Then*

*(a) If* **all** *$g_1, \ldots, g_k$ are everywhere defined on $\boldsymbol{x}$,* **and** *$\phi$ is everywhere defined on $\boldsymbol{u}$, then $f$ is everywhere defined on $\boldsymbol{x}$.*

*(b) If* **all** *$g_1, \ldots, g_k$ are everywhere defined and continuous on $\boldsymbol{x}$,* **and** *$\phi$ is everywhere defined and continuous on $\boldsymbol{u}$, then $f$ is everywhere defined and continuous on $\boldsymbol{x}$.*

*(c) If* **all** *$g_1, \ldots, g_k$ are bounded on $\boldsymbol{x}$,* **and** *$\phi$ maps bounded sets in $\mathbb{R}^k$ to bounded sets in $\mathbb{R}$, then $f$ is bounded on $\boldsymbol{x}$.*

*(d) If* **any** *of $g_1, \ldots, g_k$ is nowhere defined on $\boldsymbol{x}$,* **or** *$\phi$ is nowhere defined on $\boldsymbol{u}$, then $f$ is nowhere defined on $\boldsymbol{x}$.*

*(e) If* **any** *of $g_1, \ldots, g_k$ has empty domain, or $\phi$ has empty domain, then $f$ has empty domain.*

[*Note. The phrase "function $g$ is everywhere continuous on set $\boldsymbol{s}$" means "the restriction of $g$ to $\boldsymbol{s}$ is continuous at each point of $\boldsymbol{s}$", not "$g$ is continuous at each point of $\boldsymbol{s}$". For example, $g(x) = \mathsf{floor}(x)$ is everywhere continuous on $\boldsymbol{s} = [0, \frac{1}{2}]$, but as a function on the whole of $\mathbb{R}$ is not continuous at $0 \in \boldsymbol{s}$.*]

*Proof.* Equation (15) means, by the definition of composing partial functions, that $f(x)$ is defined and equals $\phi(u)$ where $u = (u_1, \ldots, u_k)$ and

$$u_i = g_i(x), \quad \text{for } i = 1, \ldots, k$$

for those $x$ such that $x$ is in $\operatorname{dom} g_i$ for each $i$ and the resulting $u$ is in $\operatorname{dom} \phi$.

Suppose the conditions of part (a) hold. Then for any $x \in \boldsymbol{x}$, $u_i \in \boldsymbol{g}_i$ by (16) and the definition of range. Hence $u \in \boldsymbol{u}$, so $\phi(u)$ is defined, so $f(x_1, \ldots, x_n)$ is defined, proving (a).

Suppose, in addition, the continuity assumptions of (b) hold. Let $(x^{(r)})_{r=1,2,\ldots}$ be a sequence of points *entirely within* $\boldsymbol{x}$ and converging to some $x \in \boldsymbol{x}$. By (b) and the definition of "continuity on $\boldsymbol{x}$", each sequence $u_i^{(r)} = g_i(x^{(r)})$, which lies entirely in $\boldsymbol{g}_i$ by (16), converges to $u_i = g_i(x)$. Then the vector sequence $u^{(r)} = (u_1^{(r)}, \ldots, u_k^{(r)})$ lies within $\boldsymbol{u}$ and converges to $u = (u_1, \ldots, u_k)$. Again by (b), $\phi(u^{(r)})$ converges to $\phi(u)$. But $\phi(u^{(r)}) = f(x^{(r)})$, and $\phi(u) = f(x)$, by the definition of $f$. Hence $f(x^{(r)})$ converges to $f(x)$, proving $f$ is continuous on $\boldsymbol{x}$ as well as defined there. This proves (b).

The other parts of the Lemma are proved similarly. □

**Lemma C.3.** *With $f = \phi \circ g$ as in Lemma C.2, let $c_j = \operatorname{dec}(g_j, \boldsymbol{x}[\operatorname{vars} g_j])$ for $j = 1, \ldots, k$, and let $c_0 = \operatorname{dec}(\phi, \boldsymbol{u})$. Then*

$$\min\{c_0, c_1, \ldots, c_k\} \supseteq \operatorname{dec}(f, \boldsymbol{x}) \tag{17}$$

*where the min is with respect to the quality ordering of decorations, and the $\supseteq$ with respect to their containment ordering.*

*Proof.* Write $d = \operatorname{dec}(f, \boldsymbol{x})$.

If $c_0, c_1, \ldots, c_k$ all equal `saf` then the hypotheses of Lemma C.2(b, c) are satisfied, whence $f$ is everywhere defined and continuous on the nonempty $\boldsymbol{x}$, and bounded on it. That is, property $p_{\mathtt{saf}}$ holds on $(f, \boldsymbol{x})$. Properties $p_{\mathtt{emp}}$ and $p_{\mathtt{ill}}$ must be false, while $p_{\mathtt{ill}}$ and $p_{\mathtt{def}}$ are weaker than $p_{\mathtt{saf}}$. By definition $d$ is the strongest value, in the containment order, for which $p_d(f, \boldsymbol{x})$ holds, so $d = \mathtt{saf}$ and (17) is satisfied.

If $\min\{c_0, c_1, \ldots, c_k\}$ is `def` then each $c_j$ is either `def` or `saf`. In either case the hypotheses of Lemma C.2(c) are satisfied, whence $f$ is everywhere defined on the nonempty $\boldsymbol{x}$. That is, property $p_{\texttt{def}}(f, \boldsymbol{x})$ holds. Arguing as in the previous paragraph we see $d$ equals either `def` or `saf` (we cannot know which). In either case $d \subseteq$ `def` and again (17) is satisfied.

If $\min\{c_0, c_1, \ldots, c_k\}$ is `con` then $d$ could have any of the five values; in all cases (17) is satisfied.

If $\min\{c_0, c_1, \ldots, c_k\}$ is `emp` then by definition either $p_{\texttt{emp}}(g_j, \boldsymbol{x}[\operatorname{vars} g_j])$ holds for at least one $i$, or $p_{\texttt{emp}}(\phi, \boldsymbol{u})$ holds. In either case the hypotheses of Lemma C.2(d) are satisfied, whence $f$ is nowhere defined on $\boldsymbol{x}$. That is, $p_{\texttt{emp}}$ holds on $(f, \boldsymbol{x})$. In fact $p_{\texttt{ill}}$ might hold. Neither of $p_{\texttt{saf}}$ or $p_{\texttt{def}}$ can hold since they require $f$ to be somewhere defined on $\boldsymbol{x}$. Arguing as before, we see $d$ equals either `emp` or `ill` (we cannot know which). In either case $d \subseteq$ `emp` and again (17) is satisfied.

If $\min\{c_0, c_1, \ldots, c_k\}$ is `ill` then by definition either $p_{\texttt{ill}}(g_j, \boldsymbol{x}[\operatorname{vars} g_j])$ holds for at least one $i$, or $p_{\texttt{ill}}(\phi, \boldsymbol{u})$ holds. In either case the hypotheses of Lemma C.2(e) are satisfied, whence $f$ has empty domain. That is, $p_{\texttt{ill}}$ holds on $(f, \boldsymbol{x})$. Arguing as before, we see $d$ equals `ill`, and again (17) is satisfied. □

**Lemma C.4.** *Let $c_0, \ldots, c_k$ and $c'_0, \ldots, c'_k$ be decorations such that $c_i \supseteq c'_i$ for $i = 0, \ldots, k$. Then*

$$\min\{c_0, \ldots, c_k\} \supseteq \min\{c'_0, \ldots, c'_k\}.$$

*That is, making the terms stronger (in the "weak sense"!) makes the minimum stronger.*

*Note that $\supseteq$ is the containment order, and the min is with respect to the quality order.*

*Proof.* Since $\supseteq$ is transitive, we can prove the result inductively by changing $c_i$ to $c'_i$ one term at a time. That is, it suffices to consider the case where $c_i \neq c'_i$ holds for just one $i$; hence, without loss, to prove that if $b \supseteq b'$, then

$$\min\{b, c_1, \ldots, c_k\} \supseteq \min\{b', c_1, \ldots, c_k\}. \tag{18}$$

Denote the left and right sides of (18) by $d$ and $d'$. Clearly, in the quality order, which is total, $d \leq d'$ or $d \geq d'$ according as $b \leq b'$ or $b \geq b'$.

If $b' =$ `con` then by the definition of the containment order, $b \supseteq b'$ implies $b =$ `con` also, and the result is trivial. Otherwise if, in the quality order, $b'$ is worse than `con`, the relation $b \supseteq b'$ implies $b$ satisfies $b' \leq b \leq$ `con`, so also $d' \leq d \leq$ `con` by the previous paragraph, which by the definition of the containment order gives $d \supseteq d'$.

Similarly if $b'$ is better than `con`, the relation $b \supseteq b'$ implies $b$ satisfies $b' \geq b \geq$ `con`, so also $d' \geq d \geq$ `con`, which again gives $d \supseteq d'$.

This proves (18) and hence the Lemma, as argued above. □

**Proof of FTDIA.** Let f be an arithmetic expression. Let it have $n$ independent variables, which without loss we label $z_1, \ldots, z_n$. Let $f$ denote the point function of f, or its decorated interval function, as appropriate. Let the actual argument $\boldsymbol{x}_b = ((\boldsymbol{x}_1, b_1), \ldots, (\boldsymbol{x}_n, b_n))$ be any decorated box in $\overline{\mathbb{DIR}}^n$. Any of the $\boldsymbol{x}_i$ may be empty.

It is required to prove that the result of decorated interval evaluation, $\boldsymbol{f}_d = f(\boldsymbol{x}_b)$, encloses the decorated range $\operatorname{drange}(f, \boldsymbol{x}_b)$ of the point function.

*Base case.* Expression f is one of the variables $z_i$. The theorem holds in this case, since it is assumed that the decorated interval version of the identity map $id(x) = x$ ($x \in \mathbb{R}$) is a decorated interval extension of $id()$, see note at end of C.1.1.

*Induction step.* Expression f has the form

$$f = \phi(g_1, \ldots, g_k).$$

where the $g_1, \ldots, g_k$ are expressions and $\phi$ is an arithmetic operation.

Let decorated box $\boldsymbol{g}_c = (\boldsymbol{g}_1, c_1), \ldots, (\boldsymbol{g}_k, c_k)$ be the result of evaluating the decorated interval function of each $g_j$ on the relevant subvector of the input decorated box:

$$(\boldsymbol{g}_j, c_j) = g_j(\boldsymbol{x}_b[\operatorname{vars} \mathrm{g}_j]), \quad j = 1, \ldots, k. \tag{19}$$

By the inductive hypothesis, this is a decorated interval extension of the point function of $g_j$, so for $j = 1, \ldots, k$

$$\boldsymbol{g}_j \supseteq \mathrm{range}(g_j, \boldsymbol{x}[\mathrm{vars}\,\mathsf{g}_j]), \tag{20}$$
$$c_j \supseteq \mathrm{dec}(g_j, \boldsymbol{x}_b[\mathrm{vars}\,\mathsf{g}_j]) \tag{21}$$
$$= \min\{b_{0j}, b[\mathrm{vars}\,\mathsf{g}_j]\}, \tag{22}$$

where $b_{0j} = \mathrm{dec}(g_j, \boldsymbol{x}[\mathrm{vars}\,\mathsf{g}_j])$ and $b[\mathrm{vars}\,\mathsf{g}_j]$ denotes the list of $b_i$ for $i \in \mathrm{vars}\,\mathsf{g}_j$.

By the hypothesis of the theorem, we compute

$$\boldsymbol{f}_d = \phi(\boldsymbol{g}_c) \tag{23}$$

with a decorated interval extension of the point version of $\phi$, so that

$$\boldsymbol{f} \supseteq \mathrm{range}(\phi, \boldsymbol{g}), \tag{24}$$
$$d \supseteq \mathrm{dec}(\phi, \boldsymbol{g}_c),$$
$$= \min\{c_0, c_1, \ldots, c_k\} \tag{25}$$

where

$$c_0 = \mathrm{dec}(\phi, \boldsymbol{g}). \tag{26}$$

It is required to prove

$$\boldsymbol{f} \supseteq \mathrm{range}(f, \boldsymbol{x}), \tag{27}$$
$$d \supseteq \mathrm{dec}(f, \boldsymbol{x}_b). \tag{28}$$

For (27), take any $y \in \mathrm{range}(f, \boldsymbol{x})$. By the definition of composing partial functions, there exists $x \in \boldsymbol{x}[\mathrm{vars}\,\mathsf{f}]$ such that $\gamma_j = g_j(x[\mathrm{vars}\,\mathsf{g}_j])$ is defined for $j = 1, \ldots, k$ (here we used $\mathrm{vars}\,\mathsf{g}_j \subseteq \mathrm{vars}\,\mathsf{f}$), and $\phi(\gamma_1, \ldots, \gamma_k)$ is defined and equals $y$. By (20), $\gamma_j \in \boldsymbol{g}_j$ for $j = 1, \ldots, k$, hence

$$y \in \mathrm{range}(\phi, \boldsymbol{g}) \qquad \text{by the definition of range}$$
$$\subseteq \boldsymbol{f} \qquad \text{by (24).}$$

Since $y$ was arbitrary this proves (27).

For (28), write

$$c'_0 = \mathrm{dec}(\phi, \boldsymbol{g}), \tag{29}$$
$$c'_j = \mathrm{dec}(g_j, \boldsymbol{x}_b[\mathrm{vars}\,\mathsf{g}_j]) \text{ for } j = 1, \ldots, k. \tag{30}$$

Then by (21) we have $c_j \supseteq c'_j$ for $j = 1, \ldots, k$ and by (26) this holds for $j = 0$ also, hence by Lemma C.4

$$\min\{c_0, c_1, \ldots, c_k\} \supseteq \min\{c'_0, c'_1, \ldots, c'_k\}. \tag{31}$$

Also, by Lemma C.3,

$$\min\{c'_0, c'_1, \ldots, c'_k\} \supseteq \mathrm{dec}(f, \boldsymbol{x}_b) \tag{32}$$

Since $\supseteq$ is transitive, combining (25, 31, 32) proves (28).

This completes the induction step and the proof. □

C.1.3. *Discussion.* ⚠ The main problem that I see is that my concept "decoration of a function over a decorated box", $\mathrm{dec}(f, \boldsymbol{x}_b)$, is faulty. It seemed attractive to derive it from "decoration of a function over the bare box $\boldsymbol{x}$" combined with the decoration vector $b$. But boundedness can't be handled by separating interval from decoration in this way.

[*Example. In some finite precision inf-sup type, let* $\boldsymbol{x}_b = [0, \mathtt{realmax}]_{\mathtt{saf}}$ *and apply* $\mathrm{sqr}(x) = x^2$. *We have* $\boldsymbol{y}_c = \mathrm{sqr}(\boldsymbol{x}_b) = [0, +\infty]_{\mathtt{saf}}$ *(OK) but then* $\boldsymbol{z}_d = \mathrm{sqr}(\boldsymbol{y}_c) = [0, +\infty]_{\mathtt{def}}$ *so the boundedness information has been lost.*]

We need a redefinition so that this loss doesn't occur. I have been trying an "intrinsic" definition of $\mathrm{dec}(f, \boldsymbol{x}_b)$ based on what seems to me a natural idea, of a **history** of $f : \mathbb{R}^n \to \mathbb{R}$ over $\boldsymbol{x}_b \in \overline{\mathbb{DIR}}^n$. This means a map $\xi : \mathrm{dom}\,\xi \subseteq \mathbb{R}^m \to \mathbb{R}^n$ and a **nonempty** box $\boldsymbol{s} \in \overline{\mathbb{IR}}^m$ such that

$$\mathrm{range}(\xi, \boldsymbol{s}) \subseteq \boldsymbol{x}$$
$$p_{b_i}(\xi_i, \boldsymbol{s}) \text{ is true for } i = 1, \ldots, n.$$

Informally, a history is a possible way by which $\boldsymbol{x}_b$ might have arisen from previous computation.

Then I want dec$(f, \boldsymbol{x}_b)$ to be the strongest decoration $d$ such that $p_d(f \circ \xi, \boldsymbol{s})$ is true for **all** histories of $f$ over $\boldsymbol{x}_b$.

This seems to nearly work, and for most cases to give the $\min\{c_0, c_1, \ldots, c_n\}$ formula. But as far as I can see there are cases where "strongest decoration" is not well defined. Can anyone help analyse this idea?

C.2. **The "Not an Interval" object.** From 4.4.6, the zero-argument function with empty domain is the real constant function with value NaN, "Not a Number". It is easily seen that NaN's strict interval extension is the interval constant function with value $\emptyset$, and its strict decorated interval extension is the decorated interval constant function with value NaI $= (\emptyset, \texttt{ill})$. (This was pointed out by Arnold Neumaier.)

The decorated interval NaI has behaviour that qualifies it for the role of "Not an Interval". By definition it signals that it is the result of evaluating a null function, with empty domain.

It is returned by any invalid call to an interval constructor, such as "the interval from 3 to NaN". It is unconditionally "sticky" within arithmetic expressions, in the sense that if any argument to an arithmetic operation is NaI, then that operation's output is NaI.

However, it cannot be generated "new" during evaluation of any expression that uses normal operations, even if the theoretical function being defined has empty domain. For example, the expression

$$f(x) = \sqrt{-1 - x^2}$$

clearly defines, over the reals, a function with empty domain; but decorated interval evaluation can *never* notice this. With any non-NaI input, it will return $(\emptyset, \texttt{emp})$ and not $(\emptyset, \texttt{ill})$.

Hence, in practice, NaI behaves as one expects it to do: it records the "taint of illegitimacy" of an interval's ancestry. A decorated interval is NaI iff it is the result of an ill-formed construction or is the computational descendant of such a result.

C.3. **Some implementation considerations.** (Informative.) ⚠ Still to be revised in the light of work by Hayes and Neumaier.

Treating BIs, BDs and DIs as all one abstract type is fairly straightforward at level 1 (and at level 2 for a given interval type). By promotion, the BIs can be identified with a subset of the DIs. This is analogous to how a real number $x$ may be identified with the point interval $[x, x]$ or the complex number $x+0i$. It is also consistent, in the sense that promoting, doing a DI operation, and taking the interval part of the result is always equivalent to doing the corresponding BI operation.

However this identification is not particularly useful. One reason is that for BDs, no consistent way exists to identify them with a subset of the DIs, so they *must* be regarded as a separate set. More important reasons are to do with storage and efficiency.

For illustration consider the interval type that is likely to be most commonly used: a 754-conforming binary64 inf-sup representation. A bare interval thus takes up 16 bytes. Sophisticated encodings aside, a general decorated interval *must* use extra space for the decoration: say a byte. Thus we have 16-byte BIs and 17-byte DIs.

Because access across the memory hierarchy is typically in chunks of 4, 8 or 16 bytes, 17-byte objects are likely to be read and written about half as fast as 16-byte ones, and might also be stored with nearly 50% waste space. Clever compiler methods may avoid this by storing the decoration byte separately, but this is not to be relied on. Hence the Level 1 datatype should be implemented to avoid 17-byte arithmetic where possible.

Some algorithms need to carry both an interval value and a decoration even after the latter becomes "not safe". They need 17-byte arithmetic. However, for various important algorithms, the interval value is of no interest after an exception occurs. The set $\{\,\mathrm{BI}\,\} \cup \{\,\mathrm{BD}\,\}$ is closed under arithmetic operations, so such algorithms are happy with an arithmetic that stores either a BI or a BD, but never both together, and does so in a 16 byte interval format.

This can be done by encoding the BDs in unused bit-patterns, of which there are many. For implementations that represent the empty set as (NaN, NaN), one possibility is to store the decoration value in the payload of an empty set. Arithmetic on the resulting interval format proceeds by doing the standard bare interval operation. If all arguments are BIs and the local status $s$ of the operation is `saf`, the interval result is returned; otherwise a decoration computed from $s$ and any BD arguments is returned.

Logic is needed to extract any argument payloads on entry, and to combine $s$ with any other decorations and choose which result to return, on exit. This apart, no extra tests or other change to the numeric operation's code are needed because payloads are invisible to floating point arithmetic operations. There is some parallelism between the numeric and decoration processing, which could make this $\{\,\mathrm{BI}\,\} \cup \{\,\mathrm{BD}\,\}$ arithmetic nearly as fast as bare interval arithmetic.

If the memory access and storage problems of 17-byte objects were overcome, full $\{\,\mathrm{BI}\,\} \cup \{\,\mathrm{DI}\,\} \cup \{\,\mathrm{BD}\,\}$ arithmetic could also be nearly as fast as bare interval arithmetic.

In summary, implementations *shall* provide BI arithmetic and $\{\,\mathrm{BI}\,\} \cup \{\,\mathrm{DI}\,\} \cup \{\,\mathrm{BD}\,\}$ arithmetic, which the standard requires. On efficiency grounds, for inf-sup interval types, they *should* provide the more limited $\{\,\mathrm{BI}\,\} \cup \{\,\mathrm{BD}\,\}$ arithmetic.

C.4. **Examples of use of decorations.**

TBW