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.

The decoration system is designed in a way that naive users of interval arithmetic do not notice anything about decorations, unless they inquire explicitly about their values. They are only required to

– call the domain operation, see 4.8.3, for the input vector of any function evaluation used to invoke an existence theorem,
– explicitly convert all floating-point constants (but not integer constants) to intervals,

and have the full rigor of interval calculations available. A smart compiler may even relieve users from these tasks.

Expert users can inspect, set and modify decorations to improve the efficiency of their code, at their own risk of having to check that the computations done in this way remain rigorously valid.

Decorations are based on the desire that, 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 in the form of 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 for inspection afterwards.

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.

4.8.2. *Definitions.* The set $\mathbb{D}$ of **decorations** has seven elements, linearly ordered by **quality**:

Value	Short description
`bnd`	bounded
`saf`	safe
`def`	defined
`con`	containing
`emp`	empty
`ill`	ill-formed
`ein`	empty input

The total ordering

$$\texttt{ein} < \texttt{ill} < \texttt{emp} < \texttt{con} < \texttt{def} < \texttt{saf} < \texttt{bnd}, \tag{5}$$

is implied whenever taking the minimum or maximum of decorations. The smallest decoration `ein` is the "worst" and the largest decoration `bnd` is the "best".

Formally, each $d \in \mathbb{D}$ represents the set of pairs $(f, \boldsymbol{x})$ consisting of a real-valued function f with domain $\mathrm{Dom}\, f \subseteq \mathbb{R}^n$ 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

$$\begin{array}{ll}
p_{\texttt{ein}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is empty.} \\
p_{\texttt{bnd}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is a subset of } \mathrm{Dom}\, f, \text{ and the restriction of } f \text{ to } \boldsymbol{x} \text{ is continuous and bounded;} \\
p_{\texttt{saf}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is a nonempty subset of } \mathrm{Dom}\, f, \text{ and the restriction of } f \text{ to } \boldsymbol{x} \text{ is continuous;} \\
p_{\texttt{def}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is a nonempty subset of } \mathrm{Dom}\, f; \\
p_{\texttt{con}}(f, \boldsymbol{x}) : & \text{always true;} \\
p_{\texttt{emp}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is nonempty and disjoint from } \mathrm{Dom}\, f \\
p_{\texttt{ill}}(f, \boldsymbol{x}) : & \boldsymbol{x} \text{ is nonempty and } \mathrm{Dom}\, f \text{ is empty.}
\end{array} \tag{6}$$

In addition to the total ordering by quality, the decorations are also partially ordered by containment between the sets just defined:

$$\texttt{ein} \subseteq \texttt{bnd} \subseteq \texttt{saf} \subseteq \texttt{def} \subseteq \texttt{con} \supseteq \texttt{emp} \supseteq \texttt{ill}; \tag{7}$$

note the reversal of the containment signs after `con`. $d \subseteq d'$, which is equivalent to $d' \supseteq d$, says that d is stronger than d', i.e., that d' is weaker than d. Thus `con` is the weakest decoration, where nothing is claimed; `ein` and `ill` claim the most and are strongest.

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 invariance property holds:

$$d \text{ is in } \{\, \texttt{bnd}, \texttt{saf}, \texttt{def}, \texttt{con} \,\} \text{ if } \boldsymbol{x} \text{ is nonempty, and in } \{\, \texttt{emp}, \texttt{ill}, \texttt{ein} \,\} \text{ if } \boldsymbol{x} \text{ is empty.} \tag{8}$$

(The other combinations would not make sense.) The set of decorated intervals is denoted by $\overline{\mathbb{DIR}}$. 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 by $\overline{\mathbb{DIR}}^n$.

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

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

$$(\boldsymbol{x}, d) \supseteq (\boldsymbol{x}', d') \quad \text{iff} \quad \boldsymbol{x} \supseteq \boldsymbol{x}' \text{ and } d \supseteq d', \tag{9}$$

using set inclusion for the first component and the order (7) 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 Theorem.* The decoration system provides an extended version of Moore's Theorem 4.1 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,

$$\mathrm{dec}(f, \boldsymbol{x}) = \begin{cases}
\texttt{ein} & \text{if } p_{\texttt{ein}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{bnd} & \text{if } p_{\texttt{ein}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{bnd}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{saf} & \text{if } p_{\texttt{bnd}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{bnd}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{def} & \text{if } p_{\texttt{saf}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{def}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{ill} & \text{if } p_{\texttt{ill}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{emp} & \text{if } p_{\texttt{ill}}(f, \boldsymbol{x}) \text{ fails and } p_{\texttt{emp}}(f, \boldsymbol{x}) \text{ holds;} \\
\texttt{con} & \text{otherwise.}
\end{cases} \tag{10}$$

This is well-defined because $p_d(f, \boldsymbol{x})$ cannot hold for both $d = \texttt{ein}$ and $d = \texttt{ill}$.

Note that though con loooks trivial on first sight, 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, $\mathrm{dec}(f, \boldsymbol{x}) = \texttt{con}$ implies that $\boldsymbol{x}$ contains infinitely many points.

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

$$\mathrm{dec}(f, \boldsymbol{x}_c) := \min\{c', c''\} \quad \text{where } c' = \min\{c_1, \ldots, c_n\}, \tag{11}$$

and $c'' = \mathrm{dec}(f, \boldsymbol{x})$ whenever $c' < \texttt{bnd}$, and the minimum is with respect to the quality order (5). That is, in this case 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. In the remaining case $c' = \texttt{bnd}$, we require instead $c'' = \min \mathrm{dec}(f, \boldsymbol{y})$ over all bounded $\boldsymbol{y} \subseteq \boldsymbol{x}$, to match the semantics of the bnd decoration.

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 $\mathrm{Drng}(f, \boldsymbol{x}_c)$, is the pair

$$\mathrm{Drng}(f, \boldsymbol{x}_c) = (\mathrm{Rng}(f, \boldsymbol{x}),\ \mathrm{dec}(f, \boldsymbol{x}_c)). \tag{12}$$

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** of f is a mapping $\boldsymbol{f}$ from decorated boxes $\boldsymbol{x}_d \in \overline{\mathbb{DIR}}^n$ to decorated intervals, such that

$$\boldsymbol{f}(\boldsymbol{x}_d) \supseteq \mathrm{Drng}(f, \boldsymbol{x}_d) \tag{13}$$

in the componentwise containment order defined in (9), for any such decorated box $\boldsymbol{x}_d$, where its interval part is regarded as a subset of $\mathbb{R}^n$.

A **decorated interval version** of an expression f is a decorated interval extension whose interval part is an interval extension of f whose value is $\emptyset_{\texttt{ein}}$ when the input box is empty, i.e., has an empty component. (The second specification is needed in order to cover correctly the case when some input variable is not present in the expression.)

The generalized fundamental theorem is as follows.

Theorem 4.2 (Fundamental Theorem of Decorated Interval Arithmetic, FTDIA)**.** *Any decorated interval version of an arithmetic expression* f *is a decorated interval extension of the point function of* f.

The rigor necessary to make interval arithmetic a reliable tool for verified computing warrants a detailed proof, given in Annex C.

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 an enclosure of the range an enclosure in the sense of (7), providing information on properties of a function being everywhere defined, continuous, etc., over a box.

To obtain maximal 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 interval or box $\boldsymbol{x}$ to a decorated interval (box) as follows:

$$\mathbf{domain}(\boldsymbol{x}) = \boldsymbol{x}_d \text{ where } d_i = \begin{cases} \texttt{ein} & \text{if } \boldsymbol{x} \text{ is empty (i.e., has some empty component),} \\ \texttt{saf} & \text{if } \boldsymbol{x}_i \text{ is unbounded and } \boldsymbol{x} \text{ is nonempty,} \\ \texttt{bnd} & \text{if } \boldsymbol{x}_i \text{ is bounded and } \boldsymbol{x} \text{ is nonempty.} \end{cases} \tag{14}$$

Note that this function acts componentwise

$$\mathbf{domain}(\boldsymbol{x}) = (\mathbf{domain}(\boldsymbol{x}_1), \ldots, \mathbf{domain}(\boldsymbol{x}_n)) \tag{15}$$

only if *all* components of $\boldsymbol{x}$ are nonempty, whereas

$$\mathbf{domain}(\boldsymbol{x}) = (\emptyset_{\texttt{emp}}, \ldots, \emptyset_{\texttt{emp}}) \tag{16}$$

if *any* component of $\boldsymbol{x}$ is empty. Note also that this function erases any potential information about the ill-formedness of a component $\boldsymbol{x}_i$.

⚠ This is AN's proposed definition. JDP thinks one should set

$$\mathbf{domain}(\boldsymbol{x}) = (\emptyset_{\texttt{ill}}, \ldots, \emptyset_{\texttt{ill}})$$

if *any* component of $\boldsymbol{x}$ is ill-formed.

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 $\boldsymbol{y}_e = \boldsymbol{f}(\boldsymbol{x}_d)$ for some decorated interval extension;
3. Inspect the decoration e.

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

(i) $d = \texttt{ein}$ iff $D = \texttt{ein}$ iff the input box is empty.

(ii) Otherwise the input box is guaranteed to be nonempty, and we have one of the following possibilities:
- If $d = \texttt{bnd}$ then D must also be `bnd`, so f is guaranteed to be everywhere defined, continuous and bounded on the nonempty box $\boldsymbol{x}$.
- If $d = \texttt{saf}$ then D must also `saf` or `bnd`, so f is guaranteed to be everywhere defined and continuous on the nonempty box $\boldsymbol{x}$. It might be bounded there as well, but this is not known.
- If $d = \texttt{def}$, then D must be `def` or `saf` or `bnd`, so f is guaranteed to be everywhere defined on the nonempty box $\boldsymbol{x}$. It might be continuous and/or bounded there as well, but this is not known.
- If $d = \texttt{ill}$, then D must be `ill`: Dom f is empty, that is, f is nowhere defined, although the box $\boldsymbol{x}$ is nonempty. This occurs if and only if some intermediate constant operation was ill-formed. This has a special significance of *Not an Interval* at the computational level: see C.2. In particular, ill-formedness is sticky in expressions evaluated on a box generated by the domain function.
- If $d = \texttt{emp}$, then D must be `emp` or `ill`: Dom f is guaranteed to be disjoint from the nonempty box $\boldsymbol{x}$. This includes the cases where $\boldsymbol{x}$ is empty, which has no special name, and where Dom f is empty, which is the `ill` case.
- If $d = \texttt{con}$, then D might be any of the decorations $\neq \texttt{ein}$, and no further 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 the decorated interval evaluation of* $f(x, y) = \sqrt{x(y-x)-1}$ *with various input intervals* $\boldsymbol{x}$, $\boldsymbol{y}$, *using the natural decorated interval extension of each arithmetic operation. The natural domain* Dom f *is easily seen to be the union of the regions* $x > 0$, $y \geq x + 1/x$ *and* $x < 0$, $y \leq x + 1/x$.

For manageable notation, we 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* $\boldsymbol{x}_{dx} = [1, 2]_{\texttt{bnd}}$, $\boldsymbol{y}_{dy} = [3, 4]_{\texttt{bnd}}$. *The first operation is*

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

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* $du' = \mathrm{dec}(-, (\boldsymbol{y}, \boldsymbol{x})) = \texttt{bnd}$, *whence by (11) the (best possible) decoration on* $\boldsymbol{u}$ *is* $du = \min\{du', dy, dx\} = \min\{\texttt{bnd}, \texttt{bnd}, \texttt{bnd}\} = \texttt{bnd}$. *Multiplication is also "nice", so the second operation similarly gives*

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

The constant 1, following 4.4.4, becomes a decorated interval function returning the constant value $[1, 1]_{\texttt{bnd}}$. *The next operation is again "nice", and gives*

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

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

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

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

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

The final result says

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

whence, $\mathsf{dec}(f, \boldsymbol{z}) = \texttt{bnd}$. *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* bnd. *However the evaluation fails to detect this because of interval widening due to the dependence problem of interval arithmetic. Namely after* $\boldsymbol{u}_{du} = [\frac{5}{2}, 3]_{\texttt{bnd}}$, $\boldsymbol{v}_{dv} = [\frac{5}{2}, 6]_{\texttt{bnd}}$, $\boldsymbol{w}_{dw} = [-\frac{1}{2}, 5]_{\texttt{bnd}}$, *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* $dw' = \mathsf{dec}(\sqrt{\cdot}, \boldsymbol{w}) = \mathsf{dec}(\sqrt{\cdot}, [-\frac{1}{2}, 5]) = \texttt{con}$ *giving* $\mathsf{dec}(\sqrt{\cdot}, \boldsymbol{w}_{dw}) = \min\{dw', dv\} = \texttt{con}$, *so finally* $\boldsymbol{f}_{df} = [0,\sqrt{5}]_{\texttt{con}}$. *This is a valid enclosure of the decorated range* $[0,\sqrt{5}]_{\texttt{bnd}}$, *but safety (though true) is not guaranteed by the computation performed.*

(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 $\mathsf{Rng}(\psi, \boldsymbol{x}) = [2, 2\frac{1}{2}]$.

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 for the above function just entails adding code to compute an enclosure of the decoration $d = \mathsf{dec}(\psi, \boldsymbol{x}_c)$ *over an input decorated interval* $\boldsymbol{x}_c$, *along the lines of the following:*

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

// combine with input decoration by equation (11):
6. $d = \min\{c_0, c\}$.

]

4.8.5. *Case expressions and case function.* Subclause 4.6.2 defined the function $\mathbf{case}(b, g, h)$ and its behavior on bare intervals. Its decorated interval version $\mathbf{case}(\boldsymbol{b}_{db}, \boldsymbol{g}_{dg}, \boldsymbol{h}_{dh})$ is defined according to the standard worst case semantics, and returns $\boldsymbol{f}_{df}$ where $df = \min\{df', db, dg, dh\}$ and

$$df' = \begin{cases} \texttt{emp} & \text{if } \boldsymbol{b} \text{ contains neither 0 nor 1} \\ & \text{(in normal use this only occurs when } \boldsymbol{b} \text{ is empty)} \\ \texttt{bnd} & \text{elseif } \boldsymbol{b} \text{ contains 0 and } \boldsymbol{h} \text{ is bounded} \\ & \text{or } \boldsymbol{b} \text{ contains 1 and } \boldsymbol{g} \text{ is bounded} \\ \texttt{saf} & \text{elseif } \boldsymbol{b} \text{ contains 0 and } \boldsymbol{h} \text{ is unbounded} \\ & \text{or } \boldsymbol{b} \text{ contains 1 and } \boldsymbol{g} \text{ is unbounded} \\ \texttt{con} & \text{otherwise.} \end{cases}$$

Note that this correctly handles empty input: In this case, one of db, dg, dh is `ein`, resulting in $df = \texttt{ein}$.

[*Note. The above handling of cases renders superfluous the use of interval comparisons or overlap relations (which are both incompatible with the decoration concept since there is no way to account for exceptions) as a mechanism for handling cases.*]

4.8.6. *Bare object arithmetic with a threshold.* Bare intervals and bare decorations are called **bare objects**. For experienced users, and for more efficient execution (see **??** ⚠ forward ref, probably to level 3), we define the arithmetic operations, and intersection and union, also on bare objects, extending the standard interval arithmetic on bare intervals.

In many applications, the use of decorations is limited to the following particular situation. The only use that is made of a function evaluation $f(x)$ at a decorated interval $\boldsymbol{x}$ (initialized as a domain) depends on a check whether the resulting decoration is $\geq T$ in the quality order where T is an application-dependent **exception threshold**. The threshold T is one of $\{\texttt{con}, \texttt{def}, \texttt{saf}, \texttt{bnd}\}$. It declares any decoration $< T$ to be an exception.

– If so, the decoration is not used, but one exploits the range enclosure given by the interval part.
– Otherwise, the interval part is not used, but one exploits the information given by the decoration.

In this case, one can implement the decoration scheme without storing explicit decorations. This is done dependent on the exception threshold; the programmer or a compiler can choose the appropriate threshold based on the final use made of the interval evaluation. The storage of bare object data, whether interval or decoration, can be done within two floating-point numbers. Thus in the kind of applications mentioned above, bare objects usually give equivalent results to full decorated intervals at less cost in storage and communication.

The promotion scheme ensures that the tightest enclosing decoration compatible with the input is returned. Hence, in an extended calculation, the claim about the result of a function evaluation using bare object arithmetic is never stronger than the result of the same evaluation using full decorated interval arithmetic. Therefore, the FTDIA continues to hold.

The results for arithmetic operations (or one of the nonarithmetic operations intersect and union) on bare objects are determined according to the following rules for **promoting** bare objects to decorated intervals, which follow necessarily if the fundamental theorem is to cover computations involving bare and/or decorated arguments.

Each nonempty bare interval is treated as decorated with decoration T, and each empty bare interval as decorated with decoration `emp`. Operations on bare intervals are performed in this mode as if they were decorated in the way described, resulting in a decorated interval $\boldsymbol{z}_d$ converted back into a bare object. Whenever a result with decoration $d < T$ is obtained, the result is recorded as the bare decoration d. Otherwise, the result is recorded as the bare interval $\boldsymbol{z}$.

For arithmetic operations with at least one bare decoration input, the result is always a bare decoration. A bare decoration d in $\{\texttt{emp}, \texttt{ill}\}$ is promoted to $\emptyset_d$, and a bare decoration d in $\{\texttt{saf}, \texttt{def}, \texttt{con}\}$ is promoted (conceptually, not algorithmically) to $\boldsymbol{x}_d$ with any nonempty $\boldsymbol{x}$. In the latter case, after performing the operation leading to the result $\boldsymbol{z}_d$, the tightest decoration (in the containment order (7)) enclosing all compatible $\boldsymbol{z}$ is returned. This analysis is done conceptually; since there are only a few decorations, one can prepare complete operation tables for the decorations according to these promotion rules, and only these tables need to be implemented. They are simplified by the fact that a bare decoration input to an operation must necessarily be $< T$.

[*** I think this table belongs to an Annex. It spells out consequences of the requirements,

TABLE 5. Bare object operations for $+, -, \times, \div$ and $\sqrt{\cdot}$ with threshold $T \in \{\texttt{con}, \texttt{def}, \texttt{saf}\}$.

Here c, d are bare decorations $< T$, and $\boldsymbol{x}, \boldsymbol{y}$ are bare intervals. Independently of T, if any input is `ill` the result is `ill`, else if any input is `emp` or the empty interval the result is `emp`. The tables below give the remaining cases where

$$\texttt{con} \le c < T,\ \texttt{con} \le d < T, \text{ and } \boldsymbol{x}, \boldsymbol{y} \text{ are nonempty.} \tag{17}$$

Binary operations, where $\boldsymbol{x}$ or c is the left operand and $\boldsymbol{y}$ or d is the right operand.

$+, -, \times$	$\boldsymbol{y}$	d
$\boldsymbol{x}$	Normal bare interval result	d
c	c	$\min(c, d)$

$\div$	$\boldsymbol{y} = [0,0]$	$0 \in \boldsymbol{y} \ne [0,0]$	$0 \notin \boldsymbol{y}$	d
$\boldsymbol{x}$	`emp`	If T>`con` then `con`, else normal bare interval result	Normal bare interval result	`con`
c	`emp`	`con`	c	`con`

Square root, where $\boldsymbol{x} = [\underline{x}, \overline{x}]$.

	case	
$\sqrt{\boldsymbol{x}}$	$\overline{x} < 0$	`emp`
	$\underline{x} < 0 \le \overline{x}$	If T>`con` then `con`, else normal bare interval result
	$\underline{x} \ge 0$	Normal bare interval result
$\sqrt{c}$		`con`

not requirements themselves, and is therefore redundant, though very useful for the implementor. Moreover, the tables need extension to handle `bnd` and to cover all required elementary functions. ***] Table 5 gives the operation tables for the four basic operations.

[*Examples. In items (b) onwards, conditions (17) are assumed.*

(a) *Justification for* $\texttt{emp} + \boldsymbol{x} = \texttt{emp}$, *independent of* T.
This promotes to $(\emptyset, \texttt{emp}) + (\boldsymbol{x}, T) = \big(\emptyset, \min(\texttt{emp}, T, \texttt{emp})\big)$. *Since* $\texttt{emp} < T$ *this equals* $(\emptyset, \texttt{emp})$ *which gives an exception (again because* $\texttt{emp} < T$*) so is recorded as the bare decoration* `emp`. *The same holds if* $+$ *is replaced by* $-, \times$ *or* $\div$.

(b) *Justification for* $\boldsymbol{x} \times d = d$ *independent of* T.
Since $\boldsymbol{x}$ *is nonempty and* $d \ge \texttt{con}$, *this promotes to* $(\boldsymbol{x}, T) \times (\boldsymbol{y}, d)$ *with arbitrary nonempty* $\boldsymbol{y}$, *giving* $\big(\boldsymbol{x} \times \boldsymbol{y}, \min(T, d, e)\big)$ *where* e *is* `saf` *if* $\boldsymbol{x} \times \boldsymbol{y}$ *is bounded, otherwise* `def`. *Now* $d < T$ *so* d *cannot exceed* `def`, *hence* $d \le e$, *so* $\min(T, d, e) = d$.

(c) *Justification for* $c/d = \texttt{con}$ *independent of* T.
Since $c, d \ge \texttt{con}$, c/d *promotes to* $(\boldsymbol{x}, c)/(\boldsymbol{y}, d)$ *with arbitrary nonempty* $\boldsymbol{x}, \boldsymbol{y}$, *giving* $\big(\boldsymbol{x}/\boldsymbol{y}, \min(c, d, e)\big)$ *where* $e = \texttt{emp}$ *if* $\boldsymbol{y} = [0,0]$, *else* $e = \texttt{con}$ *if* $0 \in \boldsymbol{y}$, *else* $e = \texttt{saf}$. *So* $\min(c, d, e) \ge \texttt{con}$ *and can equal* `con`, *so the tightest enclosing decoration is* `con`.

(d) *Justification for* $\boldsymbol{x}/\boldsymbol{y}$ *when* $0 \in \boldsymbol{y} \ne [0,0]$.
$\boldsymbol{x}/\boldsymbol{y}$ *promotes to* $(\boldsymbol{x}, T)/(\boldsymbol{y}, T)$ *giving* $\big(\boldsymbol{x}/\boldsymbol{y}, \min(T, T, \texttt{con})\big) = (\boldsymbol{x}/\boldsymbol{y}, \texttt{con})$. *If* $T > \texttt{con}$ *this gives an exception so the decoration* `con` *is returned; if* $T = \texttt{con}$ *it is not an exception, so the interval* $\boldsymbol{x}/\boldsymbol{y}$ *is returned.*

(e) *Justification for* $\sqrt{\boldsymbol{x}}$ *with* $\boldsymbol{x} = [\underline{x}, \overline{x}]$ *and* $\underline{x} < 0 \le \overline{x}$.
$\sqrt{\boldsymbol{x}}$ *promotes to* $\sqrt{(\boldsymbol{x}, T)}$, *giving* $(\sqrt{\boldsymbol{x}}, \texttt{con})$ *which in the given case equals* $([0, \sqrt{\overline{x}}], \texttt{con})$. *As with the previous item, if* $T > \texttt{con}$ *then* `con` *is returned; if* $T = \texttt{con}$ *then* $\sqrt{\boldsymbol{x}}$ *is returned.*

]