d. Not every interval encoding necessarily encodes an interval object, but when it does, that object is unique. Each interval object has at least one encoding and may have more than one.

5. Flavors

5.1. Flavors overview. The standard permits different interval *flavors*, which embody different foundational (Level 1) approaches to intervals. An implementation shall provide at least one flavor. For brevity, phrases such as "A flavor shall provide, or document, a feature" mean that the implementation of that flavor shall provide the feature, or its documentation describe it.

Flavor is a property of program execution context, not of an individual interval, therefore just one flavor shall be in force at any point of execution. It is recommended that at the language level, the flavor should be constant at the level of a procedure/function, or of a compilation unit.

A flavor is identified by a unique name. Certain flavors, termed **included**, are specified in this standard. The *(list to be confirmed)* flavors are the currently included flavors. The procedure for submitting a new flavor for inclusion is described in Annex B. A conforming implementation that provides one or more included flavors may also provide non-included flavors, without losing conformance for the included flavors.

The flavor concept enforces a common core of behavior that different kinds of interval arithmetic must share:

(i) The set of required operations, identified by their names, is the same in all flavors. Similarly the set of recommended operations is the same in all flavors. See §**??**, **??**.
(ii) There is a set of *common intervals* whose members are—in a sense made precise below—intervals of any flavor.
(iii) There is a set of common evaluations of library operations, with common intervals as input, that give—again in a sense made precise below—the same result in any flavor.

The result in item (iii) is a mathematically tightest (Level 1) result, ignoring any interval widening due to finite precision (Level 2).

5.2. Definition of common intervals and common evaluations. The choice of the set of common intervals, and the set of common evaluations of an operation, is a design decision that defines the flavor concept. It should aim for simplicity, and the common evaluations should be specified by a general rule that makes it easy to add a new operation to the library if needed. The choice that was made is specified in the following paragraphs.

All likely flavors extend the classical Moore arithmetic [**4**] on the set $\mathbb{IR}$ of *closed bounded nonempty real intervals*, and no other intervals belong to all of them. Hence, the chosen set $\mathfrak{C}$ of common intervals is $\mathbb{IR}$.

The common evaluations are specified in terms of graphs of interval operations. For an interval operation φ of arity k, its graph (in some flavor) is a subset of a $(k+1)$-dimensional space of intervals, namely the set of interval $(k+1)$-tuples $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ such that $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}$ is true in that flavor. Each such tuple is called an *operation instance*.

The general rule is that each φ has a set $\mathfrak{CE}(\varphi)$ of **common evaluations**: operation instances $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ such that all its components are in $\mathbb{IR}$ and

$$\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y} \qquad \text{shall hold in all flavors.}$$

$\mathfrak{CE}(\varphi)$ may be regarded as the *flavor-independent graph* of φ. For brevity, writing

$$\text{the evaluation } \varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y} \text{ is common,} \tag{1}$$

or the equivalent notation when φ is an infix operator (e.g., $\boldsymbol{x}_1 + \boldsymbol{x}_2 = \boldsymbol{y}$), means that $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y}) \in \mathfrak{CE}(\varphi)$.

The standard defines $\mathfrak{CE}(\varphi)$ as follows.

Arithmetic operation: that is, an interval extension of the corresponding point function φ. The common operation instances are those $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ such that the point function φ is *defined and continuous at each point of* the closed, bounded, nonempty box $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$, and $\boldsymbol{y}$ equals the range of φ over this box. Then necessarily $\boldsymbol{y}$ belongs to $\mathbb{IR}$.

Non-arithmetic operation: The common operation instances are those tuples with common inputs $\boldsymbol{x}_i$ such that *the result* $\boldsymbol{y}$ *is also common.* In particular for `convexHull`, the common operation instances are those $(\boldsymbol{x}_1, \boldsymbol{x}_2;\ \boldsymbol{y})$ with arbitrary $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{IR}$ and $\boldsymbol{y}$ equal to the convex hull of $\boldsymbol{x}_1 \cup \boldsymbol{x}_2$. For `intersection`, they are those $(\boldsymbol{x}_1, \boldsymbol{x}_2;\ \boldsymbol{y})$ with arbitrary $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{IR}$ and $\boldsymbol{y}$ equal to $\boldsymbol{x}_1 \cap \boldsymbol{x}_2$, provided the latter is *nonempty* (since $\emptyset \notin \mathbb{IR}$).

[*Examples.*

- *The evaluation* $[-1,4]/[3,4] = [-1/3, 4/3]$ *is common; but* $[3,4]/[-1,4] = \boldsymbol{y}$ *is not common, for any* $\boldsymbol{y} \in \mathbb{IR}$.
- *In the set-based flavor,* $\sqrt{[-1,4]} = [0,2]$. *But in the Kaucher flavor* $\sqrt{[-1,4]}$ *is undefined, so* $\sqrt{[-1,4]} = \boldsymbol{y}$ *cannot be common for any* $\boldsymbol{y}$. *In fact* $\mathfrak{CE}$(`sqrt`) *is the set of all* $([a,b];\ [\sqrt{a}, \sqrt{(b)}])$ *for which* $0 \le a \le b < +\infty$.
- *Similarly,* $([-1,4] \cap [5,6] = \emptyset)$ *in the set-based flavor, while* $([-1,4] \cap [5,6] = [5,4])$ *in the Kaucher flavor. Thus* $([-1,4] \cap [5,6] = \boldsymbol{y})$ *cannot be common for any* $\boldsymbol{y}$.
- *The above definition for arithmetic operations requires "*φ *is defined and continuous at each point of* $\boldsymbol{x}$*", which is weaker (may give a smaller set* $\mathfrak{CE}(\varphi)$*) than "the restriction of* φ *to* $\boldsymbol{x}$ *is everywhere defined and continuous". E.g., the common evaluations of the function* floor(x) *are all* $([a,b];\ [k,k])$ *with* $k \in \mathbb{Z}$, $k < a \le b < k+1$. *Thus* floor$([1, 1.9]) = [1,1]$ *is not common, because* floor() *is not continuous at* 1, *despite its restriction to* $[1, 1.9]$ *being everywhere continuous.*

]

5.3. Loose common evaluations. At Level 2, common evaluations are usually not computable because of roundoff; instead, an enclosing interval of some finite precision interval type is computed. The notion of a **loose common evaluation** of an operation φ takes account of this: it is defined to be any

$$\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}' \tag{2}$$

where $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}$ is common and $\boldsymbol{y}'$ is a member of $\mathbb{IR}$ containing $\boldsymbol{y}$. A member of $\mathfrak{CE}(\varphi)$ may be called **tight**, to emphasize that it is not loose. The set of loose common evaluations is uniquely determined by the set of (tight) common evaluations.

Informally, for a given φ and $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$, the loose common evaluations describe all closed bounded intervals that might be produced by evaluating an enclosure of $\mathrm{Rge}(\varphi \,|\, \boldsymbol{x})$ in finite precision. [*Note. Different qualities of enclosure are distinguished by the terms* tightest *(same as tight in this context),* accurate *and* valid *introduced in §9.10.*]

5.4. Relation of common evaluations to flavors. The formal definition of common evaluations takes into account that the common intervals are not necessarily a subset of the intervals of a given flavor, but are identified with a subset of it by an embedding map.

[*Examples.*

A Kaucher interval is defined to be a pair (a, b) *of real numbers—equivalently, a point in the plane* $\mathbb{R}^2$*—which for* $a \le b$ *is "proper" and identified with the normal real interval [a,b], and for* $a > b$ *is "improper". Thus the embedding map is* $\boldsymbol{x} \mapsto (\inf \boldsymbol{x}, \sup \boldsymbol{x})$ *for* $\boldsymbol{x} \in \mathbb{IR}$.

For the set-based flavor, every common interval is actually an interval of that flavor ($\mathbb{IR}$ *is a subset of* $\overline{\mathbb{IR}}$*), so the embedding is the identity map* $\boldsymbol{x} \mapsto \boldsymbol{x}$ *for* $\boldsymbol{x} \in \mathbb{IR}$.]

Formally, a flavor is identified by a pair $(\mathfrak{F}, \mathfrak{f})$ where $\mathfrak{F}$ is a set of Level 1 entities, the *intervals* of that flavor, and $\mathfrak{f}$ is a one-to-one *embedding map* $\mathbb{IR} \to \mathfrak{F}$. Usually, $\mathfrak{f}(\boldsymbol{x})$ is abbreviated to $\mathfrak{f}\boldsymbol{x}$.

It is then required that *operation compatibility* shall hold for each library operation φ and for each flavor $(\mathfrak{F}, \mathfrak{f})$. Namely, given $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ and $\boldsymbol{y}$ in $\mathbb{IR}$,

$$\begin{array}{l}\text{If } (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y}) \text{ is a common operation instance of } \varphi, \\ \text{then } (\mathfrak{f}\boldsymbol{x}_1, \mathfrak{f}\boldsymbol{x}_2, \ldots, \mathfrak{f}\boldsymbol{x}_k;\ \mathfrak{f}\boldsymbol{y}) \text{ is an operation instance of } \varphi \text{ in flavor } \mathfrak{F}.\end{array} \tag{3}$$

That is, if the evaluation $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}$ is common, then $\varphi(\mathfrak{f}\boldsymbol{x}_1, \mathfrak{f}\boldsymbol{x}_2, \ldots, \mathfrak{f}\boldsymbol{x}_k)$ must be defined in $\mathfrak{F}$ with value $\mathfrak{f}\boldsymbol{y}$.

An evaluation in $\mathfrak{F}$ of an expression, in which only (loose) common evaluations of elementary operations occur, is called a common evaluation of that expression. That is, in a flavor $(\mathfrak{F}, \mathfrak{f})$, the expression's inputs are members of $\mathfrak{f}(\mathbb{IR})$, and each intermediate value is produced by a common evaluation of an operation so that it is also in $\mathfrak{f}(\mathbb{IR})$; hence the final result is in $\mathfrak{f}(\mathbb{IR})$.

The com decoration makes it possible to determine, for a specific expression and specific interval inputs, whether common evaluation has occurred, see Clause 6.

5.5. Flavors and the Fundamental Theorem. Suppose f is an arithmetic expression (does not contain set operations such as intersection) so that it defines a real point function $f(x_1, \ldots, x_n)$, and is evaluated in classical interval arithmetic over a box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ where the $\boldsymbol{x}_i$ are in $\mathbb{IR}$. The classical FTIA (§2.3) can be stated in the following form:

> If the evaluation is common then f is everywhere defined and continuous on $\boldsymbol{x}$,
> and the output $\boldsymbol{y}$ encloses the range of f over $\boldsymbol{x}$.

This holds mathematically (at Level 1) using tight evaluations of individual operations in the expression, and also in finite precision (Level 2), using loose evaluations.

It also is true, modulo the embedding map, in any flavor. That is, suppose inputs $\boldsymbol{x}_j^* \in \mathfrak{F}$ are given to the expression, and evaluation gives output $\boldsymbol{y}^* \in \mathfrak{F}$. Suppose the inputs are common intervals and it is known, via the decoration system or otherwise, that the evaluation was common. Then one can map back to corresponding $\boldsymbol{x}_j \in \mathbb{IR}$ and $\boldsymbol{y} \in \mathbb{IR}$, and draw the conclusions of the classical FTIA.

[*Note. Besides this "minimal" FTIA for any flavor, which derives automatically from the classical FTIA and the meaning of common evaluation, each of the set-based and Kaucher flavors has a more general FTIA that contains the minimal FTIA as a special case. Note the Kaucher flavor has a generalized meaning of the "contains" relation, which is used in stating its FTIA:* $[a, b] \supseteq [c, d]$ *means* $(a \leq c \wedge b \geq d)$*, whether* $[a, b]$ *and* $[c, d]$ *are proper or improper. Since the minimal FTIA, above, is defined by mapping back to common intervals, it does not need to use such generalized notions.*]

It is useful to consider when a *flavor-independent result* of common evaluation occurs. That is, when does evaluating a given expression, over the "same" bounded nonempty box in two different flavors, give identical results modulo the embedding map? At Level 1, this is always true if the individual operation evaluations are tight. Also, an implementation may create conditions under which it holds in finite precision, by the flavors "sharing" at Level 2 as follows:

- It provides some shared interval types, which represent exactly the same finite set of common intervals in each flavor.
- It provides some shared library operations (on the shared types), which when acting on common intervals have identical rounding behavior in each flavor, modulo the embedding map.

Then a common evaluation that only uses shared types and operations gives identical results in both flavors, modulo the embedding map. More details are in §**??**.

6. Decoration system

6.1. Decorations overview. A decoration is information attached to an interval; the combination is called a decorated interval. Interval calculation has two main objectives:

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

Traditional interval analysis targets the first objective; decorated intervals, as defined in this standard, target the second.

A decoration primarily describes a property, not of the interval it is attached to, but of the function defined by some code that produced the interval by evaluating over some input box.

For instance, if a section of code defines the expression $\sqrt{x^2 + xy - 1}$, then decorated-interval evaluation of this code with suitably initialized input intervals $\boldsymbol{x}, \boldsymbol{y}$ gives information about the definedness, continuity, etc. of the point function $f(x, y) = \sqrt{x^2 + xy - 1}$ over the box $(\boldsymbol{x}, \boldsymbol{y})$ in the plane.

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 only need

- call the newDec operation on the inputs of any function evaluation used to invoke an existence theorem,
- explicitly convert relevant floating-point constants (but not integer parameters such as the p in $\texttt{pown}(x, p) = x^p$) to intervals,

and have the full rigor of interval calculations available. A smart implementation may even relieve users from these tasks. Expert users can inspect, set and modify decorations to improve code

efficiency, but are responsible for checking that computations done in this way remain rigorously valid.

Especially in the set-based flavor, decorations are based on the desire that, from an interval evaluation of a real function f on a box $\boldsymbol{x}$, one should get not only a range enclosure $f(\boldsymbol{x})$ but also a guarantee that the pair $(f, \boldsymbol{x})$ has certain important properties, such as $f(x)$ being defined for all $x \in \boldsymbol{x}$, f restricted to $\boldsymbol{x}$ being continuous, etc. This goal is achieved, in parts of a program that require it, by performing *decorated interval evaluation*, 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.

In some flavors, certain interval operations ignore decorations, i.e., give undecorated interval output. Users are responsible for the appropriate propagation of decorations by these operations.

The function f is assumed to be expressed by code, an algebraic formula, etc.—generically termed an *expression*—which can be evaluated in several modes: point evaluation, interval evaluation, or decorated interval evaluation. The standard does not specify a definition of "expression"; however, Annex C gives formal proofs in terms of a particular definition, and indicates how this relates to expressions in some programming languages.

The P1788 decoration model, in contrast with 754's, has no status 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 while evaluating f, that can be inspected afterwards. This enables a fully local handling of exceptional conditions in interval calculations—important in a concurrent computing environment.

An implementation may provide any of the following: (i) status flags that are raised in the event of certain decoration values being produced by an operation; (ii) means for the user to specify that such an event signals an exception, and to invoke a system- or user-defined handler as a result. [*Example. The user may be able to specify execution be terminated if an arithmetic operation is evaluated on a box that is not wholly inside its domain—an interval version of 754's "invalid operation" exception.*] Such features are language- or implementation-defined.

6.2. Decoration definition and propagation. Each flavor shall document its set of provided decorations and their mathematical definitions. These are flavor-defined, with the exception of the decoration `com`, see §6.3.

The implementation makes the decoration system of each flavor available to the user via *decorated interval extensions* of relevant library operations. Such an operation φ, with interval inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ carrying decorations $dx_1, \ldots, dx_k$, shall compute the same interval output $\boldsymbol{y}$ as the corresponding bare interval extension of φ—hence dependent on the $\boldsymbol{x}_i$ but not on the dx_i. It shall compute a *local decoration* d, dependent on the $\boldsymbol{x}_i$ and possibly on $\boldsymbol{y}$, but not on the dx_i. It shall combine d with the dx_i by a flavor-defined *propagation rule* to give an output decoration dy, and return $\boldsymbol{y}$ decorated by dy.

The local decoration d may convey purely Level 1 information—e.g., that φ is everywhere continuous on the box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. It may convey Level 2 information related to the particular finite-precision interval types being used—e.g., that $\boldsymbol{y}$, though mathematically a bounded interval, became unbounded by overflow. For diagnostic use it may convey Level 3 or 4 information, e.g., how an interval is represented, or how memory is used.

If f is an expression, decorated interval evaluation of an expression means evaluation of f with decorated interval inputs and using decorated interval extensions of the expression's library operations. Those inputs generally need to be given suitable initial decorations that lead to the most informative output-decoration. A flavor shall provide a `newDec` function for this purpose. If $\boldsymbol{x}$ is a bare interval, `newDec`$(\boldsymbol{x})$ equals $\boldsymbol{x}$ with such an initial decoration. If $\boldsymbol{x}$ is a decorated interval, the decoration is discarded and `newDec` applied to the bare interval part.

It is the responsibility of each flavor to document the meaning of its decorations, and the correct use of these decorations within programs.

6.3. Recognizing common evaluation. A flavor may provide the decoration com with the following propagation rule for library arithmetic operations. In an implementation with more than one flavor, each flavor shall do so.

In the following, φ denotes an arbitrary interval extension of a point library arithmetic operation φ, provided by the implementation at Level 2 (typically the one associated with a particular interval type).

> Let φ applied to input intervals $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ give the computed result $\boldsymbol{y}$, and let $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}$ be a loose common evaluation as defined in (2). If each of the inputs $\boldsymbol{x}_i$ is decorated com, then the output $\boldsymbol{y}$ shall be decorated com.

Informally, com records that the individual operation φ took bounded nonempty input intervals and produced a bounded (necessarily nonempty) output interval. This can be interpreted as indicating "overflow did not occur". Further, the propagation rule ensures that if the initial inputs to an arithmetic expression f are bounded and nonempty, and are initialized with the decoration com, then the final result $\boldsymbol{y} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ is decorated com if and only if the evaluation of the whole expression was common as defined in §5.4.

Flavors should define other decoration values, but com is the only one that is required to have the same meaning in all flavors.

[*Examples. Reasons why an individual evaluation of φ with common inputs $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ may* not *return* com *include the following.*

Outside domain: *The implementation finds φ is not defined and continuous everywhere on $\boldsymbol{x}$. Examples: $\sqrt{[-4, 4]}$, $\operatorname{sign}([0, 2])$.*

Overflow: *The Level 1 result is too large to be represented. Example: Consider an interval type $\mathbb{T}$ whose intervals are represented by their lower and upper bounds in some floating point format, let* REALMAX *be the largest finite number in that format, and $\boldsymbol{x}$ be the common $\mathbb{T}$-interval $[0, \texttt{REALMAX}]$. Then $\boldsymbol{x} + \boldsymbol{x}$ cannot be enclosed in a common $\mathbb{T}$-interval.*

Cost: *It is too expensive to determine whether the result is too large to be represented. A possible example is $\tan([a, b])$ where $[a, b]$ is of an interval type $\mathbb{T}$ as in the previous item, and one of its endpoints is happens to be very close to a singularity of $\tan(x)$.*

]

7. Conformance requirements

To be completed later.