7. Flavors

7.1. Flavors overview. The standard permits different interval behaviors via the flavor concept described in this clause. An **interval model** means a particular foundational approach to interval arithmetic, in the sense of a Level 1 abstract data type of entities called intervals and of operations on them. A **flavor** is an interval model that conforms to the core specification described below.

A **provided flavor** is a flavor that the implementation provides in finite precision, see §7.5.

A **standard flavor** is one that has a specification in this standard, which extends the core specification. The *(list to be confirmed)* flavors are the currently standard flavors. The procedure for submitting a new flavor for inclusion is described in Annex A.

An implementation shall provide at least one standard flavor. The implementation as a whole is conforming if each standard flavor conforms to the specification of that flavor, and each non-standard flavor conforms to the core specification.

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 over a procedure/function or a compilation unit.

For brevity, phrases such as "A flavor shall provide, or document, a feature" mean that an implementation of that flavor shall provide the feature, or its documentation describe it.

The core specification of a flavor is summarized in the list below, and detailed in the following subclauses. The entities in the list are part of the Level 1 mathematical theory unless said otherwise.

(i) There is a flavor-independent set of *common intervals*, defined in §7.2.
(ii) There is a flavor-independent set of named *common library operations*; see Clause 9.
(iii) There is a flavor-independent set of *common evaluations* of library operations, defined in §7.3. They have common intervals as input and, in the sense of §7.2, give the same result in any flavor.
(iv) There is a flavor-defined set of *intervals* of the flavor that shall, in the sense defined in §7.2, contain the common intervals as a subset. There is a flavor-defined finite set of *decorations* as specified in Clause 8; in particular it includes the `com` decoration. A decorated interval is a pair (interval, decoration).
(v) For each library operation, a flavor shall fully specify the interval version and, if provided, decorated interval version in that flavor. That is, it shall specify the exact set of inputs (the domain) where the operation-version is defined, and the value at each such input.
(vi) There is a flavor-defined partial order called *contains* for the intervals, see §7.2, which for common intervals coincides with normal set containment.
(vii) At Level 2, intervals are organized into *types*. An (interval) type is essentially a finite set of Level 1 intervals; see §7.5.1.
(viii) The relation between an operation at Level 1 and a version of it at Level 2 is summarized as follows. The latter evaluates the Level 1 operation on the Level 1 values denoted by its inputs. If (at those inputs) the operation has a value, this is converted to a Level 2 result, of an interval type, or of a numeric format, or a boolean. An interval result may overflow, causing an exception to be signaled. If the operation has no value, an exception is signaled, or some default value returned, or both. The details are flavor-defined, see §7.5.3.

7.2. Flavor basic properties. A flavor is described at Level 1 by a set $\mathfrak{F}$ of entities, the **intervals** of the flavor, a set of decorations, and a set of named operations that includes the required operations of Clause 9. The symbol $\mathfrak{F}$ is also used to refer to the flavor as a whole.

The (flavor-independent) **common intervals** are defined to be the set $\mathbb{IR}$ of nonempty closed bounded real intervals used in classical Moore arithmetic [**6**].

The relation of $\mathfrak{F}$ to the common intervals is described by a one-to-one **embedding map** $\mathfrak{f} : \mathbb{IR} \to \mathfrak{F}$. Usually, $\mathfrak{f}(\boldsymbol{x})$ is abbreviated to $\mathfrak{f}\boldsymbol{x}$. The set of all $\mathfrak{f}\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{IR}$ forms the **common intervals of** $\mathfrak{F}$. Usually $\boldsymbol{x}$ is identified with $\mathfrak{f}\boldsymbol{x}$ and $\mathbb{IR}$ is treated as a subset of $\mathfrak{F}$, for every flavor. To emphasize that this has been done, a statement may be said to hold *modulo the embedding map*.

The set of decorations of $\mathfrak{F}$ is finite, and includes the `com` decoration; see Clause 8.

[*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* $\boldsymbol{x} \in \mathbb{IR}$.

]

For each flavor $\mathfrak{F}$, a relation $\supseteq$, called **contains** or **encloses**, shall be defined between intervals. It shall be a partial order: $\boldsymbol{x} \supseteq \boldsymbol{y}$ and $\boldsymbol{y} \supseteq \boldsymbol{z}$ imply $\boldsymbol{x} \supseteq \boldsymbol{z}$, and $\boldsymbol{x} = \boldsymbol{y}$ if and only if both $\boldsymbol{x} \supseteq \boldsymbol{y}$ and $\boldsymbol{y} \supseteq \boldsymbol{x}$ hold. For common intervals it shall have the normal meaning modulo the embedding map: if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{IR} \subseteq \mathfrak{F}$ then $\boldsymbol{x} \supseteq \boldsymbol{y}$ in $\mathfrak{F}$ if and only if $\boldsymbol{x} \supseteq \boldsymbol{y}$ in the sense of set containment.
[*Example. In the Kaucher flavor,* (a, b) *is defined to contain* (a', b') *if and only if* $a \le a'$ *and* $b \ge b'$*; e.g.,* $[1, 2] \supseteq [2, 1]$ *is true. For common (proper) intervals, this coincides with normal containment.*]

7.3. Common evaluations.

For each Level 1 version of a library operation φ that has bare intervals in its inputs or output, a set of k-tuples $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is specified for which $\varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ shall have the same Level 1 value $\boldsymbol{y}$ in all flavors. Then $\boldsymbol{x}$ is a **common input**, and $\boldsymbol{y}$ is the **common value** of φ at $\boldsymbol{x}$. The $(k+1)$-tuple $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ is a **common evaluation** of φ, alternatively written $\boldsymbol{y} = \varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. It may be called a common evaluation **instance** to emphasize that a specific input tuple is involved. The $\boldsymbol{x}_i$ and $\boldsymbol{y}$ may be of other datatypes besides intervals: real, boolean or string.

A tuple $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ whose components $\boldsymbol{x}_i$ are common intervals—nonempty closed bounded intervals in $\mathbb{R}$—can be regarded as a nonempty closed bounded box in $\mathbb{R}^k$. A tuple whose $\boldsymbol{x}_i$ are common intervals of some flavor $\mathfrak{F}$ can be identified with such a box, modulo the embedding map.

There are two cases:

(a) If φ is one of the arithmetic operations in §9.1, its point version is a real function $y = \varphi(x_1, \ldots, x_k)$ of $k \ge 0$ real variables[7]. The common inputs are those boxes $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that each $\boldsymbol{x}_i$ is common and the point version of φ is defined and continuous at each point of $\boldsymbol{x}$. The common value $\boldsymbol{y}$ is the range

$$\boldsymbol{y} = \mathrm{Rge}(\varphi \,|\, \boldsymbol{x}) = \{\, \varphi(x_1, \ldots, x_n) \mid x_i \in \boldsymbol{x}_i \text{ for each } i \,\}, \qquad \text{(here } \varphi \text{ is the point version).} \tag{3}$$

By theorems of real analysis, $\boldsymbol{y}$ is nonempty, closed, bounded and connected, so it is a common interval.

(b) If φ is one of the non-arithmetic operations in §9.2 to §9.6, it has a direct definition unrelated to a point function. Its common inputs comprise those $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that each $\boldsymbol{x}_i$ that is of interval datatype is a common interval, and $\boldsymbol{y} = \varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is defined and, if of interval datatype, is a common interval. The common value is $\boldsymbol{y}$.

7.4. Level 1 interval versions.

Let $\mathfrak{F}$ be a flavor.

7.4.1. *Bare interval operations.* For each operation φ in §9.1 to §9.6, $\mathfrak{F}$ shall define a unique Level 1 bare interval $\mathfrak{F}$*-version*. In the $\mathfrak{F}$-version, for an arithmetic operation in §9.1, each real input or output of the point operation becomes an interval of $\mathfrak{F}$; and for the operations in §9.2 to §9.6, each interval input or output becomes an interval of $\mathfrak{F}$.

The $\mathfrak{F}$-version shall be *fully specified* and shall *extend* the common evaluations. That is, the flavor shall define the set of input tuples $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, with $\boldsymbol{x}_i$ of appropriate datatypes, at which the operation is defined, and the output value $\boldsymbol{y}$ in each case; the resulting set of $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k; \boldsymbol{y})$ shall include the common evaluations of φ modulo the embedding map. Equivalently, the flavor shall define the *graph* of the $\mathfrak{F}$-version, and this graph shall contain the set of common evaluations.

Full specification makes it possible to test objectively whether, in a flavor $\mathfrak{F}$, an implemented version of an operation encloses the Level 1 result (if an interval), approximates it (if numeric) or equals it (if boolean), whenever the Level 1 result is defined in $\mathfrak{F}$.

[7] $\mathtt{pown}(x, p) = x^p$ is treated as a family of functions $\varphi_p(x)$ of real x, parameterized by the integer p. Similarly for any function having some non-real arguments.

7.4.2. *Decorated interval operations.* For each arithmetic operation φ, the flavor shall also define a Level 1 decorated $\mathfrak{F}$-version, in which each real input or output of the point operation becomes a decorated interval of $\mathfrak{F}$.

This $\mathfrak{F}$-version shall be fully specified: namely, the flavor shall define the set of decorated interval input tuples at which the operation is defined, and the decorated interval output in each case. If $\boldsymbol{y} = \varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is a common evaluation of the interval version, then $\boldsymbol{y}_{\texttt{com}} = \varphi((\boldsymbol{x}_1)_{\texttt{com}}, \ldots, (\boldsymbol{x}_k)_{\texttt{com}})$ shall be an evaluation of the decorated interval $\mathfrak{F}$- version. It is termed a Level 1 **decorated common evaluation** of φ, and $((\boldsymbol{x}_1)_{\texttt{com}}, \ldots, (\boldsymbol{x}_k)_{\texttt{com}})$ is a Level 1 **decorated common input**.

[*Note. Apart from the decorated common evaluations the relation between the bare and decorated interval versions of an operation φ is flavor-defined, but barring special cases such as "Not an Interval" input, the interval part of the latter should equal the former. That is, if φ is defined at decorated interval inputs $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$, with decorated interval value $\boldsymbol{Y}$, and if the interval parts of $\boldsymbol{X}_i$ and $\boldsymbol{Y}$ are $\boldsymbol{x}_i$ and $\boldsymbol{y}$, then $\varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ should in most cases be defined and have value $\boldsymbol{y}$.*]

7.4.3. *Flavor-specific operations.* Besides the operations required in all flavors, a flavor may define *flavor-specific* operations that are required in each implementation of that flavor. Each flavor-specific operation shall be fully specified at Level 1 by the flavor.

[*Examples.*

- *A flavor might define a decorated interval version of a non-arithmetic operation.*
- *A flavor might define an operation on decorations, adapted to its decoration model.*
- *The required operations of Clause 9 include the interval constructors* `numsToInterval` *and* `textToInterval`*. A flavor will in general extend these beyond the common evaluations. E.g., in the set-based flavor* `numsToInterval`(l, u) *constructs unbounded intervals by allowing non-common inputs with $l = -\infty$ and/or $u = +\infty$; in a Kaucher flavor it might construct improper intervals by allowing $l > u$.*
- *A flavor might provide other constructors, or give extra inputs to the required constructors. E.g., in a flavor that provides open and half-open as well as closed intervals, optional extra input(s) to* `numsToInterval` *might specify which endpoints are open.*

]

7.5. The relation of Level 1 to Level 2.

Let $\mathfrak{F}$ be a flavor.

7.5.1. *Types.* At Level 2, bare intervals of $\mathfrak{F}$ shall be organized into finite sets called (bare interval) **types**. A type is an abstraction of a particular way to represent intervals. What types are provided by an implementation is both flavor-defined and language- or implementation-defined.

There shall be a one-one correspondence between bare interval types and decorated interval types (*bare types* and *decorated types* for short). An element of a decorated type is a pair $(\boldsymbol{x}, dx)$ where $\boldsymbol{x}$ belongs to the corresponding bare type, and dx belongs to the finite set of decorations of the flavor.

An implementation shall ensure that the decoration `com` is applied only to common intervals; a common interval may however have a decoration other than `com`.

Level 2 entities are called **datums**. A type may contain some exceptional datums, e.g., a "Not an Interval" datum, besides those that denote intervals. A **$\mathbb{T}$-datum** means a general member of a type $\mathbb{T}$; a **$\mathbb{T}$-interval** means a $\mathbb{T}$-datum that denotes an interval. If the type has no exceptional datums, these terms are synonymous.

Each Level 2 bare interval shall denote a unique Level 1 interval $\boldsymbol{x}$ of $\mathfrak{F}$ and is normally regarded as being that interval. However $\mathfrak{F}$ may formally define it as a pair $(\boldsymbol{x}, t)$ where t (the type name) is a symbol that uniquely identifies the type, thus making datums of different types different even if they denote the same Level 1 interval.

Hence, each Level 2 decorated interval denotes a unique Level 1 decorated interval of $\mathfrak{F}$ but may formally be a triple $(\boldsymbol{x}, dx, t)$ of interval, decoration and type name.

7.5.2. *Hull.* Each bare type $\mathbb{T}$ has a **$\mathbb{T}$-hull** operation. At Level 1 it shall be defined by an algorithm that maps an arbitrary interval $\boldsymbol{x}$ of $\mathfrak{F}$ to a minimal $\mathbb{T}$-interval $\boldsymbol{y}$ such that $\boldsymbol{y} \supseteq \boldsymbol{x}$ in the flavor's "contains" order, if such a $\boldsymbol{y}$ exists, and otherwise is undefined. Minimal means that if $\boldsymbol{z}$ is another $\mathbb{T}$-interval and $\boldsymbol{y} \supseteq \boldsymbol{z} \supseteq \boldsymbol{x}$ then $\boldsymbol{y} = \boldsymbol{z}$.

At Level 2, an implementation should provide the $\mathbb{T}$-hull for each supported bare type $\mathbb{T}$, as an operation `convertType` that maps an arbitrary interval of any other supported bare type of $\mathfrak{F}$ to its $\mathbb{T}$-hull if this exists, and signals the flavor-independent `IntvlOverflow` exception otherwise. Its action on non-interval datums is flavor-defined.

7.5.3. *Level 2 operations.* For each bare or decorated interval version of an arithmetic operation in §9.1 and for each operation in §9.2 to §9.6, a $\mathbb{T}$-*version* of φ shall be provided for each bare interval type $\mathbb{T}$ of $\mathfrak{F}$. For each flavor-specific operation, the flavor shall give a rule to determine when an implementation shall provide a $\mathbb{T}$-version.

When passing from a Level 1 operation to a $\mathbb{T}$-version—whether required in all flavors or flavor-specific:

- Inputs or output of bare interval datatype change to type $\mathbb{T}$.
- Those of decorated interval datatype change to the decorated type of $\mathbb{T}$.
- Those of real datatype change to a type-dependent Level 2 number format $\mathbb{F}$. The flavor should make recommendations or requirements on the relation between $\mathbb{F}$ and $\mathbb{T}$.

In the description below, a *special value* of the output means a flavor-defined datum that either has diagnostic value, or is regarded as useful in practice.
[*Example. For the* `mid`() *function at Level 2 in the set-based flavor, returning the midpoint of* Empty *as* NaN*, and of* Entire *as* 0*, illustrate special values. Both values are undefined at Level 1. It was considered that no numeric value of* `mid`($\emptyset$) *makes sense, but that some algorithms are simplified by returning a default value* 0 *for* `mid`(Entire).]

For some operations and inputs the implementation might be unable to effectively compute some value or determine whether some statement is true or false—e.g., owing to constraints on time, space or algorithmic complexity. Below, the term *is found* means that the implementation is able to compute at least one such value or determine such a fact; *is not found* means the opposite.
[*Examples.*

– A constructor in the set-based flavor needs to ensure $l \le u$ when forming an interval $[l, u]$. This may be hard to check, for some implementations and types.

– In a flavor where it is a fatal error to evaluate an arithmetic operation outside its domain, it may be hard to decide if the required operation $\varphi(x) = \tan(x)$ is defined on $\boldsymbol{x} = [\underline{x}, \overline{x}]$ when $\underline{x}, \overline{x}$ are large and differ by less than π. Let format $\mathbb{F}$ be IEEE754 `binary64` *and $\mathbb{T}$ the inf-sup type based on $\mathbb{F}$. The integer $a = 214112296674652$ differs from $136308121570117\pi/2$ by less than* 2.6e–16*; and $a - 1, a, a + 1$ are exact $\mathbb{F}$-numbers. One of the $\mathbb{T}$-intervals $[a - 1, a]$ and $[a, a + 1]$ (but not both) contains a singularity of φ; but which?*]

Evaluation of a $\mathbb{T}$-version $\varphi_{\mathbb{T}}$ of an operation φ shall be as if the following is done. Let $X = (X_1, \ldots, X_k)$ be the tuple of Level 1 values denoted by the inputs to $\varphi_{\mathbb{T}}$.

1. If the Level 1 operation is found to be undefined at X then $\varphi_{\mathbb{T}}$ signals the flavor-independent `UndefinedOperation` exception, or returns a special value, or both—the choice is flavor-defined. This includes the case where some input has no Level 1 value, e.g., a numeric input is NaN.
2. If it is not found whether the Level 1 operation is defined at X then $\varphi_{\mathbb{T}}$ signals the flavor-independent `PossiblyUndefinedOperation` exception, or returns a special value, or both—the choice is flavor-defined.
3. Otherwise, it is found that the Level 1 value $Y = \varphi(X_1, \ldots, X_k)$ is defined in $\mathfrak{F}$.
   (a) If Y is a bare interval $\boldsymbol{y}$ there are two cases.
      – If a $\mathbb{T}$-interval $\boldsymbol{z}$ containing $\boldsymbol{y}$ is found (in particular if Entire exists in $\mathfrak{F}$ and is a $\mathbb{T}$-interval), then $\varphi_{\mathbb{T}}$ returns such a $\boldsymbol{z}$.
      – Otherwise, $\varphi_{\mathbb{T}}$ signals the flavor-independent `IntvlOverflow` exception and the returned result, if any, is flavor-defined.
   (b) If Y is a decorated interval $\boldsymbol{y}_{dy}$ there are three cases.
      – If φ is an arithmetic operation, and $(X_1, \ldots, X_k)$ is a decorated common input (this implies $dy =$ `com`, see §7.4.2), and a common $\mathbb{T}$-interval $\boldsymbol{z}$ containing $\boldsymbol{y}$ is found, then $\varphi_{\mathbb{T}}$ returns such a $\boldsymbol{z}$ with the decoration `com`.
      – Otherwise, if a $\mathbb{T}$-interval $\boldsymbol{z}$ containing $\boldsymbol{y}$ is found (in particular if Entire exists in $\mathfrak{F}$ and is a $\mathbb{T}$-interval), then $\varphi_{\mathbb{T}}$ returns such a $\boldsymbol{z}$ with a flavor-defined decoration.
      – Otherwise, no such $\boldsymbol{z}$ is found. Then $\varphi_{\mathbb{T}}$ signals the `IntvlOverflow` exception and the returned result, if any, is flavor-defined.

(c) If Y is numeric, $\varphi_{\mathbb{T}}$ returns a value of an appropriate number format, approximating Y in a type- and operation-dependent way.
(d) If Y is boolean, $\varphi_{\mathbb{T}}$ returns Y.

7.5.4. *Measures of accuracy.* Two **accuracy modes** for an interval-valued operation are defined for all flavors. An accuracy mode is in the first instance a property of an individual evaluation of an operation φ, over an input box $\boldsymbol{x}$, returning a result of type $\mathbb{T}$. If the evaluation does not have an interval value at Level 1, its accuracy mode is undefined. The basic property of enclosure in the sense of the flavor, defined in §7.5.3 and required for conformance, is called *valid* accuracy mode. The property that the result equals the $\mathbb{T}$-hull of the Level 1 value is called *tightest* accuracy mode.

A flavor may define other accuracy modes, and may make requirements on the accuracy achieved by an implementation. To simplify the user interface, modes should be linearly ordered by strength, with *tightest* the strongest and *valid* the weakest, where mode M is stronger than mode M' if M implies M'.

7.6. Flavors and the Fundamental Theorem (informative). For a common evaluation of an arithmetic expression, each library operation is, modulo the embedding map, defined and continuous on its inputs so that it satisfies the conditions of the strongest, "continuous" form of the FTIA in Theorem 6.1. At Level 1, the range enclosure obtained by a common evaluation is the same, independent of flavor.

It is possible in principle for an implementation to make this true also at Level 2 by providing shared number formats and interval types that represent the same sets of reals or intervals in each flavor; and library operations on these types and formats that have identical numerical behavior in each flavor. For example, both set-based and Kaucher flavors might use intervals stored as two IEEE754 `binary64` numbers representing the lower and upper bounds, and might ensure that operations, when applied to the intervals recognized by both flavors, behave identically. Such shared behavior might be useful for testing correctness of an implementation.

Beyond common evaluations, versions of the FTIA in different flavors can be strictly incomparable. For example, the set-based FTIA handles unbounded intervals, which the classical Kaucher flavor does not; conversely, Kaucher intervals have an extended FTIA applicable to reverse-bound intervals, which has no simple interpretation in the set-based flavor.

8. Decoration system

8.1. 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 real-valued functions of real variables;
– verifying the assumptions of existence, uniqueness, or nonexistence theorems.

Traditional interval analysis targets the first objective; the decoration system, as defined in this standard, targets 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.

The function f is assumed to be represented by an expression in the sense of Clause 6. For instance, if a section of code defines the expression $\sqrt{y^2-1}+xy$, 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 $f(x,y) = \sqrt{y^2-1}+xy$ 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. For example, in the set-based flavor, 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 might even relieve users from these tasks. Expert users can inspect, set and modify decorations to improve code