7. Flavors

7.1. Flavors overview. The standard permits different interval behavior 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 §7.6.

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

An implementation shall provide at least one included flavor. The implementation as a whole is conforming if each included flavor conforms to the specification of that flavor, and each non-included 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 as follows, and detailed in the following subclauses.

(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 §7.2. They have common intervals as input and, in the sense of §7.3, give the same Level 1 result in any flavor.
(iv) There is a flavor-defined set of *Level 1 intervals* that shall, in the sense defined in §7.3, contain the common intervals as a subset. There is a flavor-defined finite set of *decorations* that shall conform to the specification in Clause 8; in particular it shall include the `com` decoration.
(v) There is a flavor-defined partial order called *contains* for the Level 1 intervals, see §7.5, which for common intervals coincides with normal set containment.
(vi) At Level 2, intervals are organized into finite sets called (interval) *types*, see §7.6.1. A member of a type $\mathbb{T}$ is called a $\mathbb{T}$-datum. Each Level 2 interval denotes a unique Level 1 interval; it may be regarded as being that interval, except that it may carry a type tag to make intervals of different types different even if they denote the same Level 1 interval.
(vii) The relation between an interval-valued operation at Level 1 and a $\mathbb{T}$-datum-valued version of it at Level 2 is as follows, see §7.6.3. The latter performs the Level 1 operation on the Level 1 values denoted by its operands, and returns a $\mathbb{T}$-datum enclosing the Level 1 result. If no such enclosing interval exists, an exception is signaled.

7.2. Definition of common intervals and common evaluations. This subclause specifies at Level 1 the set of common intervals and the rules that determine the common evaluations for each of the required and recommended operations of the standard listed in Clause 9.

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

Each operation may be written $\boldsymbol{y} = \varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$, where each of the formal arguments $\boldsymbol{x}_i$ and result $\boldsymbol{y}$ is of a specified datatype—one of interval, real, boolean or string—and at least one is of interval datatype.

The rules define those k-tuples $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ for which $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ shall have the same value $\boldsymbol{y}$ in all flavors. Then $\boldsymbol{x}$ is called a **common input** and $\boldsymbol{y}$ is the **common value** of φ at $\boldsymbol{x}$. The $(k+1)$-tuple $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ is a **common evaluation**; it may be called a common evaluation **instance** to emphasize that a specific tuple of inputs is involved.

Each interval argument in a common evaluation is a common interval; to simplify description, by convention all non-interval argument instances are called common—e.g., any value of p in the integer power function $\texttt{pown}(\boldsymbol{x}, p)$.

Two cases are distinguished:

– φ is an interval *arithmetic operation*, which is an interval extension (Defn 4.2.34), of the corresponding point function $\widehat{\varphi}$. Its inputs $\boldsymbol{x}_i$ are all intervals[7] so that $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ is a box, regarded as a subset of $\mathbb{R}^k$.

The common inputs shall comprise those $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ such that each $\boldsymbol{x}_i$ is common, and $\widehat{\varphi}$ is defined and continuous at each point of $\boldsymbol{x}$. The common value $\boldsymbol{y}$ shall be the range

$$\boldsymbol{y} = \mathrm{Rge}(\widehat{\varphi} \,|\, \boldsymbol{x}) = \{\, \widehat{\varphi}(x_1, \ldots, x_n) \mid x_i \in \boldsymbol{x}_i \text{ for each } i \,\}.$$

Thus the common evaluations comprise a restriction to a subset of the set of all possible boxes of the *natural* interval extension of $\widehat{\varphi}$. Necessarily, by theorems of real analysis, $\boldsymbol{y}$ is nonempty, closed, bounded and connected, so it is a common interval.

– In all other cases, φ has a direct definition unrelated to a point function. The common inputs shall comprise those $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ such that each $\boldsymbol{x}_i$ is common, and $\boldsymbol{y} = \varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ is (a) defined and (b) if of interval datatype, is a common interval. The common value shall be $\boldsymbol{y}$.

[*Examples.*

1. *For interval division $\boldsymbol{x}_1/\boldsymbol{x}_2$ (an arithmetic operation), the common inputs are the $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ with $\boldsymbol{x}_i \in \mathbb{IR}$ and $0 \notin \boldsymbol{x}_2$.*
2. *For interval square root $\sqrt{\boldsymbol{x}}$ (an arithmetic operation), the common inputs are the $\boldsymbol{x} = [\underline{x}, \overline{x}] \in \mathbb{IR}$ for which $0 \le \underline{x} \le \overline{x} < +\infty$.*
3. *For $\mathtt{intersection}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1 \cap \boldsymbol{x}_2$ (an interval-valued non-arithmetic operation), with inputs $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that are common intervals, the result is a common interval iff it is nonempty. Hence, the common inputs are the $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ with $\boldsymbol{x}_i \in \mathbb{IR}$ and $\boldsymbol{x}_1 \cap \boldsymbol{x}_2 \neq \emptyset$. For $\mathtt{convexHull}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathsf{hull}(\boldsymbol{x}_1 \cup \boldsymbol{x}_2)$, all $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ with $\boldsymbol{x}_i \in \mathbb{IR}$ are common inputs.*
4. *The midpoint function $\mathtt{mid}(\boldsymbol{x})$ (a real-valued non-arithmetic operation) has the formula $\mathtt{mid}(\boldsymbol{x}) = (\underline{x} + \overline{x})/2$ for every common interval $\boldsymbol{x} = [\underline{x}, \overline{x}]$. Thus every common interval is a common input.*
5. *For $\mathtt{pown}(\boldsymbol{x}, p)$, the interval version of the integer power function (an arithmetic operation in its first argument, with p regarded as a parameter), the common inputs depend on the sign of p. If $p \ge 0$ they are all $(\boldsymbol{x}, p)$ with common $\boldsymbol{x}$. If $p < 0$ they are all $(\boldsymbol{x}, p)$ with common $\boldsymbol{x}$ such that $0 \notin \boldsymbol{x}$.*
6. *The above definition for arithmetic operations requires R1 "$\widehat{\varphi}$ is defined and continuous at each point of $\boldsymbol{x}$", which is a* stronger *constraint (results in fewer common evaluations) than R2 "the restriction of $\widehat{\varphi}$ to $\boldsymbol{x}$ is everywhere defined and continuous". R1 produces a* weaker *constraint on what interval arithmetics can be flavors (with fewer rules, more arithmetics obey them). In particular, requirement R1 permits cset arithmetic to be a flavor, while R2 does not.*

E.g., the common inputs for (the interval extension of) $\mathsf{floor}(x)$ *are all nonempty intervals that are disjoint from $\mathbb{Z}$. Thus* $\mathsf{floor}([1, 1.9]) = [1, 1]$ *is not common, because* $\mathsf{floor}()$ *is not continuous at 1, despite its restriction to $[1, 1.9]$ being everywhere continuous. If it were required to be common, cset arithmetic could not be a flavor.*

]

7.3. Common evaluations in a flavor. 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 $\mathfrak{f}\boldsymbol{x}$. To simplify description, by convention $\mathfrak{f}$ applied to a non-interval (real, boolean or string) value $\boldsymbol{x}$ leaves it unchanged, $\mathfrak{f}\boldsymbol{x} = \boldsymbol{x}$.

[7] Non-interval arguments such as the p of $\mathtt{pown}(\boldsymbol{x}, p)$ are treated as parameters; see Example 5.

It is then required that *operation compatibility* holds 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 $\mathbb{IR}$,

> If $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k;\ \boldsymbol{y})$ is a common evaluation instance of φ,
> then $(\mathfrak{f}\boldsymbol{x}_1, \mathfrak{f}\boldsymbol{x}_2, \ldots, \mathfrak{f}\boldsymbol{x}_k;\ \mathfrak{f}\boldsymbol{y})$ shall be an evaluation instance of φ in flavor $\mathfrak{F}$. (3)

That is, if the evaluation $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) = \boldsymbol{y}$ is common in $\mathbb{IR}$, then $\varphi(\mathfrak{f}\boldsymbol{x}_1, \mathfrak{f}\boldsymbol{x}_2, \ldots, \mathfrak{f}\boldsymbol{x}_k)$ shall be defined in $\mathfrak{F}$ with value $\mathfrak{f}\boldsymbol{y}$; it is then called a **common evaluation in $\mathfrak{F}$**.

In the rest of this clause, the map $\mathfrak{f}$ is omitted and $\mathbb{IR}$ is treated as a subset of $\mathfrak{F}$.

7.4. Non-common evaluations in a flavor. Outside the common evaluations, the Level 1 action of a library operation is flavor-defined, but it shall be a *partial function*. That is, if inputs $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ of the appropriate datatypes are given, where any interval inputs belong to the flavor $\mathfrak{F}$, then $\varphi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ is either undefined, or has a unique Level 1 value in the flavor.
[*Note. Generally the Level 1 value, if an interval, is defined to be the tightest of various possible values, in the "contains" order of the flavor.*

For example, $[1,2]/[-1,1]$ *has common inputs but is not a common evaluation. In the classical Moore flavor and in the Kaucher flavor it is undefined. In the set-based flavor its unique Level 1 value is* Entire*, being the tightest interval that contains the exact range* $\{\, x/y \mid x \in [1,2],\ y \in [-1,1],\ \text{and } y \neq 0 \,\} = \{\, z \mid -\infty < z \leq -1 \text{ or } 1 \leq z < +\infty \,\}$.

$[1,2] \cap [3,4]$ *has common inputs but is not a common evaluation. In the classical Moore flavor it is undefined. In the set-based flavor its unique Level 1 value is* Empty*. In the Kaucher flavor its unique Level 1 value is the improper interval* $[3,2]$.]

7.5. Containment. 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: 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* $a \leq a'$ *and* $b \geq b'$*; e.g.,* $[1,2] \supseteq [2,1]$ *is true. For proper intervals, this coincides with normal containment.*]

7.6. The relation of Level 1 to Level 2.

7.6.1. *Types.* At Level 2, bare intervals shall be organized into finite sets called (interval) **types**. (For decorated intervals see §8.) 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.

Level 2 entities are called **datums**. A flavor may define some special values, 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 is not a special value. If the type has no special values, these terms are synonymous.

Each Level 2 interval denotes a unique Level 1 interval $\boldsymbol{x}$ and is normally regarded as being that interval. However a flavor 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.

7.6.2. *Hull.* Each 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 the flavor to a minimal $\mathbb{T}$-datum $\boldsymbol{y}$ such that $\boldsymbol{y} \supseteq \boldsymbol{x}$ in the flavor's "contains" order, if such a $\boldsymbol{y}$ exists, and otherwise is undefined.

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

7.6.3. *Level 2 operations.* The relation between an operation φ at Level 1 and a $\mathbb{T}$-datum-valued version $\varphi_{\mathbb{T}}$ of it at Level 2 shall be as follows. If any input is a special datum, the result is flavor-defined. Otherwise, $\varphi_{\mathbb{T}}$ performs the Level 1 operation (§7.4) on the Level 1 values—of interval, or other, datatype—denoted by its operands.

– Any error that occurs, apart from that in the next item, is handled in a flavor-defined way.
– If $\boldsymbol{y}$ is an interval, $\varphi_{\mathbb{T}}$ returns a $\mathbb{T}$-interval enclosing $\boldsymbol{y}$. If no such enclosing $\mathbb{T}$-interval can be computed, $\varphi_{\mathbb{T}}$ signals the flavor-independent `ContainmentFailure` exception; the returned result, if any, is flavor-defined.

– If $\boldsymbol{y}$ is numeric, $\varphi_{\mathbb{T}}$ returns a flavor-defined approximation to $\boldsymbol{y}$.
– If $\boldsymbol{y}$ is a boolean or string, $\varphi_{\mathbb{T}}$ returns $\boldsymbol{y}$.

An evaluation, for which both the inputs and the Level 2 result are common, is called a **Level 2 common evaluation** (or a **finite-precision** common evaluation). An evaluation of an expression, in which only Level 2 common evaluations occur, is a Level 2 **common evaluation of the expression** in the relevant flavor.

[*Note. For an arithmetic expression, this is a finite-precision evaluation where the expression's inputs are in* $\mathbb{IR}$*, and the result of each of its operations, including the final result, is in* $\mathbb{IR}$*.*

The `com` *decoration makes it possible to determine whether common evaluation of an arithmetic expression has occurred, see §8.3.*]

7.6.4. *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, defined in §7.6.3 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. Modes shall 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'.

7.7. Flavors and the Fundamental Theorem. For a common evaluation of an arithmetic expression, each library operation is (i.e., can be regarded as, modulo the embedding map) defined and continuous on its inputs so that it satisfies the conditions of the strongest, "continuous" form of the FTIA, Theorem 6.1. At Level 1, using the tightest interval extension of each operation, the range enclosure obtained by a common evaluation is (again modulo the embedding map) 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 Kaucher flavor does not; while Kaucher intervals have an extended FTIA, involving generalized meanings of "contains" and "interval extension" 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