9.4. Numeric functions of intervals

The operations in Table 9.2 are defined for all common intervals, with the formula shown.

TABLE 9.2. Required numeric functions of intervals.

Name	Definition
$\inf(\boldsymbol{x})$	$\underline{x}$
$\sup(\boldsymbol{x})$	$\overline{x}$
$\mathrm{mid}(\boldsymbol{x})$	$(\underline{x}+\overline{x})/2$
$\mathrm{wid}(\boldsymbol{x})$	$\overline{x}-\underline{x}$
$\mathrm{rad}(\boldsymbol{x})$	$(\overline{x}-\underline{x})/2$
$\mathrm{mag}(\boldsymbol{x})$	$\sup\{\,\lvert x\rvert \mid x \in \boldsymbol{x}\,\} = \max(\lvert\underline{x}\rvert,\, \lvert\overline{x}\rvert)$
$\mathrm{mig}(\boldsymbol{x})$	$\inf\{\,\lvert x\rvert \mid x \in \boldsymbol{x}\,\} = \begin{cases} \min(\lvert\underline{x}\rvert,\, \lvert\overline{x}\rvert) & \text{if } \underline{x},\overline{x} \text{ have the same sign} \\ 0 & \text{otherwise} \end{cases}$

9.5. Boolean functions of intervals

The comparison relations in Table 9.3 have a boolean (1 = true, 0 = false) result.

TABLE 9.3. Comparisons for intervals $\boldsymbol{a}$ and $\boldsymbol{b}$. Notation $\forall_a$ means "for all a in $\boldsymbol{a}$", and so on. Column 4 gives formulae when $\boldsymbol{a}=[\underline{a},\overline{a}]$ and $\boldsymbol{b}=[\underline{b},\overline{b}]$ are common.

Name	Symbol	Defining predicate	Common $\boldsymbol{a},\boldsymbol{b}$	Description
`equal`$(\boldsymbol{a},\boldsymbol{b})$	$\boldsymbol{a}=\boldsymbol{b}$	$\forall_a \exists_b\, a=b \wedge \forall_b \exists_a\, b=a$	$\underline{a}=\underline{b} \wedge \overline{a}=\overline{b}$	$\boldsymbol{a}$ equals $\boldsymbol{b}$
`subset`$(\boldsymbol{a},\boldsymbol{b})$	$\boldsymbol{a}\subseteq\boldsymbol{b}$	$\forall_a \exists_b\, a=b$	$\underline{b}\le\underline{a} \wedge \overline{a}\le\overline{b}$	$\boldsymbol{a}$ is a subset of $\boldsymbol{b}$
`interior`$(\boldsymbol{a},\boldsymbol{b})$	$\boldsymbol{a}\Subset\boldsymbol{b}$	$\forall_a \exists_b\, a<b \wedge \forall_a \exists_b\, b<a$	$\underline{b}<\underline{a} \wedge \overline{a}<\overline{b}$	$\boldsymbol{a}$ is interior to $\boldsymbol{b}$
`disjoint`$(\boldsymbol{a},\boldsymbol{b})$	$\boldsymbol{a}\cap\!\!\!\!/\,\boldsymbol{b}$	$\forall_a \forall_b\, a\ne b$	$\overline{a}<\underline{b} \vee \overline{b}<\underline{a}$	$\boldsymbol{a}$ and $\boldsymbol{b}$ are disjoint

9.6. Operations on/with decorations

The function `newDec` adds a decoration to a bare interval $\boldsymbol{x}$:

$$\texttt{newDec}(\boldsymbol{x}) = \boldsymbol{x}_d$$

where d depends on $\boldsymbol{x}$ and the flavor, such that the result is suitable input for decorated-interval evaluation of an expression in that flavor.

For a decorated interval $\boldsymbol{x}_d$, the operations `intervalPart`$(\boldsymbol{x}_d)$ and `decorationPart`$(\boldsymbol{x}_d)$ have value $\boldsymbol{x}$ and d, respectively.

9.7. All-flavor interval and number literals

This subclause defines a flavor-independent syntax of literals for common intervals, which may be extended by a flavor to include its non-common intervals.

9.7.1. Overview

An **interval literal** of a flavor is a (text) string that denotes a Level 1 interval of the flavor. It is a **bare interval literal** or a **decorated interval literal** according as it denotes a bare or a decorated interval. A **number literal** is a string that denotes an extended-real number; a (decimal) **integer literal** is a particular case. A **decoration literal** is an alphanumeric string that denotes a decoration; these shall be in one to one correspondence with the decorations of the flavor. The string `com` shall denote the decoration `com` in all flavors.

Bare and decorated interval literals are used as input to bare and decorated versions of `textToInterval` in 9.8. In this standard, number literals are only used within interval literals. The definitions of literals are not intended to constrain the syntax and semantics that a language might use to denote numbers and intervals in other contexts.

The value of an interval literal is a bare or decorated Level 1 interval $\boldsymbol{x}$. Level 2 operations with interval literal inputs are evaluated following 7.5.3; typically they return the $\mathbb{T}$-hull of $\boldsymbol{x}$ for some interval type $\mathbb{T}$. *[Example. The interval denoted by the bare literal* `[1.2345]` *is the Level 1 single-point bare interval* $\boldsymbol{x} = [1.2345, 1.2345]$. *However, the result of* $\mathbb{T}$`-textToInterval("[1.2345]")`*, where* $\mathbb{T}$ *is the IEEE 754 infsup* `binary64` *type of the set-based flavor, is the interval, approximately* $[1.2344999999999999, 1.2345000000000002]$*, whose bounds are the nearest* `binary64` *numbers on either side of* 1.2345*.]*

An **all-flavor** literal is one that has the same value (modulo the embedding map if it is an interval literal) in all flavors. An all-flavor interval literal denotes a common interval; if decorated it has the decoration `com`. Given a flavor, a **literal of the flavor** is one that shall be supported in each implementation of the flavor; thus such literals include the all-flavor literals.

An implementation may support an extended form of literals, e.g., using number literals in the syntax of the host language of the implementation. It may restrict the support of literals at Level 2, by relaxing conversion accuracy of hard cases: rational number literals, long strings, etc. It shall document such extensions and restrictions.

A **common interval literal** is one that denotes a common bare or decorated interval. It might be all-flavor, of the flavor or implementation-defined.

The case of alphabetic characters in interval and number literals is ignored (e.g., `[1,1e3]_com` is equivalent to `[1,1E3]_COM`.) By default number syntax shall be that of the default locale (C locale); locale-specific variants may be provided.

9.7.2. All-flavor number literals

A sign is a plus sign `+` or a minus sign `-`. An integer literal comprises an optional sign and (i.e., followed by) a nonempty sequence of decimal digits, with the usual integer value. It is called unsigned if the sign is absent. A positive-natural literal is an unsigned integer literal whose value is not zero.

An all-flavor number literal denotes a real number. It has one of the following forms.

a) A decimal number. This comprises an optional sign, a nonempty sequence of decimal digits optionally containing a point, and an optional exponent field comprising `e` and an integer literal exponent. The value of a decimal number is the value of the sequence of decimal digits with optional point multiplied by ten raised to the power of the value of the exponent, negated if there is a leading minus sign.

b) A number in the hexadecimal-floating-constant form of the C99 standard (ISO/IEC9899, N1256 (6.4.4.2)), equivalently hexadecimal-significand form of IEEE Std 754-2008 (5.12.3). This comprises an optional sign, the string `0x`, a nonempty sequence of hexadecimal digits optionally containing a point, and an exponent field comprising `p` and an integer literal exponent. The value of a hexadecimal number is the value of the sequence of hexadecimal digits with optional point multiplied by two raised to the power of the value of the exponent, negated if there is a leading minus sign.

c) A rational literal p `/` q. This comprises an integer literal p, the `/` character, and a positive-natural literal q. Its value is the value of p divided by the value of q.

9.7.3. Unit in last place

The "uncertain form" of interval literal, below, uses the notion of the *unit in the last place* of a number literal $\boldsymbol{s}$ of some radix b, possibly containing a point but without an exponent field. Ignoring the sign and any radix-specifying code (such as `0x` for hexadecimal), $\boldsymbol{s}$ is a nonempty sequence of radix-b digits optionally containing a point. Its *last place* is the integer $p = -d$ where $d = 0$ if $\boldsymbol{s}$ contains no point, otherwise d is the number of digits after the point. Then $\mathtt{ulp}(\boldsymbol{s})$ is defined to equal b^p. When context makes clear, "x ulps of $\boldsymbol{s}$" or just "x ulps" means $x \times \mathtt{ulp}(\boldsymbol{s})$. *[Example. For the decimal strings* `123` *and* `123.`*, as well as* `0` *and* `0.`*, the last place is* 0 *and one ulp is* 1. *For* `.123` *and* `0.123`*, as well as* `.000` *and* `0.000`*, the last place is* -3 *and one ulp is* 0.001.*]*

Table 9.4. All-flavor bare interval literal examples.

Form	Literal	Exact value
Inf-sup	`[1.e-3, 1.1e-3]`	$[0.001, 0.0011]$
	`[-0x1.3p-1, 2/3]`	$[-19/32, 2/3]$
	`[3.56]`	$[3.56, 3.56]$
Uncertain	`3.56?1`	$[3.55, 3.57]$
	`3.56?1e2`	$[355, 357]$
	`3.560?2`	$[3.558, 3.562]$
	`3.56?`	$[3.555, 3.565]$
	`3.560?2u`	$[3.560, 3.562]$
	`-10?`	$[-10.5, -9.5]$
	`-10?u`	$[-10.0, -9.5]$
	`-10?12`	$[-22.0, 2.0]$

9.7.4. All-flavor bare interval literals

An all-flavor bare interval literal has one of the following forms. To simplify stating the needed constraints, e.g., $l \leq u$, the number literals l, u, m, r are identified with their values.

a) Inf-sup form: A string `[` l `,` u `]` where l and u are all-flavor number literals with $l \leq u$. Its common bare value is the common interval $[l, u]$. A string `[` m `]` with all-flavor number literal m is equivalent to `[` m `,` m `]`.

b) Uncertain form: a string m `?` r u E where: m is a decimal number literal of form a) in 9.7.2, without exponent; r is empty or is a non-negative integer literal *ulp-count*; u is empty or is a *direction character*, either `u` (up) or `d` (down); and E is empty or is an *exponent field* comprising the character `e` followed by an integer literal *exponent* e. No whitespace is permitted within the string.

With `ulp` meaning $\texttt{ulp}(m)$, the literal m? by itself denotes m with a symmetrical uncertainty of half an ulp, that is the interval $[m - \frac{1}{2}\texttt{ulp}, m + \frac{1}{2}\texttt{ulp}]$. The literal $m?r$ denotes m with a symmetrical uncertainty of r ulps, that is $[m - r \times \texttt{ulp}, m + r \times \texttt{ulp}]$. Adding `d` (down) or `u` (up) converts this to uncertainty in one direction only, e.g., $m?\texttt{d}$ denotes $[m - \frac{1}{2}\texttt{ulp}, m]$ and $m?r\texttt{u}$ denotes $[m, m + r \times \texttt{ulp}]$. The exponent field if present multiplies the whole interval by 10^e, e.g., $m\,?r\texttt{u}\,\texttt{e}e$ denotes $10^e \times [m, m + r \times \texttt{ulp}]$.

[Examples. Table 9.4 illustrates all-flavor bare interval literals. These strings are not all-flavor bare interval literals: `[1_000_000]`, `[1.0 e3]`, `[1,2!comment]`, `[2,1]`, `[5?1]`, *@5 ?1@*, `5??u`, `[]`, `[empty]`, `[ganz]`, `[1,]`, `[1,inf]`*.]*

9.7.5. All-flavor decorated interval literals

An all-flavor decorated interval literal is a string $\boldsymbol{s}$ comprising a bare interval literal $\boldsymbol{sx}$ and a decoration literal $\boldsymbol{sd}$, separated by an underscore "`_`". If, in the flavor, $\boldsymbol{sx}$ and $\boldsymbol{sd}$ have the values $\boldsymbol{x}$ and d respectively and $\boldsymbol{x}_d$ is a permitted combination (e.g., for the set-based flavor see 11.4) then $\boldsymbol{s}$ is a decorated interval literal in the flavor, with value $\boldsymbol{x}_d$.

9.7.6. Grammar for all-flavor literals

The syntax of all-flavor integer and number literals and of all-flavor bare and decorated interval literals is defined by integerLiteral, numberLiteral, bareIntvlLiteral and decoratedIntvlLiteral, respectively, in the grammar in Table 9.5. An all-flavor literal of any of these four kinds is a string that after conversion to lowercase is accepted by this grammar.

9.8. Constructors

An interval constructor by definition is an operation that creates a bare or decorated interval from non-interval data. Constructors of bare intervals are defined in each flavor as follows.