7. Levels 3 and 4 description

This clause is about Level 3, where Level 2 datums are represented, and operations on them described, and about Level 4 requirements for interchange representation and encoding.

Level 3 entities are called *objects*—they represent Level 2 datums and may be referred to as *concrete*, while the datums are *abstract*. Each Level 2 (abstract) library operation is implemented by a corresponding Level 3 (concrete) operation, whose behavior shall be consistent with the abstract operation.

7.1 Representation

The property that defines a representation is:

> Each interval datum shall be represented by at least one object. Each object shall represent at most one interval datum.

[Examples. Three possible representations are:

inf-sup form. Any interval $\boldsymbol{x}$ is represented at Level 3 by the object $(\texttt{inf}(x), \texttt{sup}(x))$ of two b64 numbers. All intervals have only one Level 3 representation because operations inf and sup are uniquely defined at Level 2 (6.7.6): interval $[0, 0]$ has representation $(-0, +0)$, interval Empty has representation $(+\infty, -\infty)$.

inf-sup-nan form. The objects are defined to be pairs (l, u) where l, u are b64 datums. A nonempty interval $\boldsymbol{x} = [\underline{x}, \overline{x}]$ is represented by an object (l, u) such that the values of l and u are $\underline{x}$ and $\overline{x}$, and Empty is represented by $(\mathrm{NaN}, \mathrm{NaN})$.

neginf-sup-nan form. This is as the previous, except that for a nonempty interval the value of l is $-\underline{x}$.

If, in these descriptions l, u and NaN are viewed as Level 2 datums, then interval $[0, 0]$ has four representatives in **inf-sup-nan** and **neginf-sup-nan** forms: $(-0, +0)$, $(-0, -0)$, $(+0, +0)$, $(+0, -0)$. Each nonempty interval with nonzero bounds has only one representative: there are unique l and u. Empty has also only one representative: there is an unique NaN. However, NaN itself has representatives, and from this viewpoint Empty has more than one representative: there are many NaNs, quiet or signaling and with different payloads, to use in $\mathrm{Empty} = (\mathrm{NaN}, \mathrm{NaN})$.]

7.2 Operations and representation

Each Level 2 (abstract) library operation is implemented by a corresponding Level 3 (concrete) operation, whose behavior shall be consistent with the abstract operation.

When an input Level 3 object does not represent a Level 2 datum, the result is implementation-defined. An implementation shall provide means for the user to specify that an `InvalidOperand` exception be signaled when this occurs.

7.3 Interchange representations and encodings

The purpose of interchange representations is to allow loss-free exchange of Level 2 interval data. This is done by imposing a standard Level 3 representation using Level 2 datums.

The standard Level 3 representation of an interval datum $\boldsymbol{x}$ is an ordered pair

$$\big(\texttt{inf}(x),\ \texttt{sup}(x)\big)$$

of two b64 datums. For example, the only representative of Empty is the pair $(+\infty, -\infty)$, and the only representative of $[0, 0]$ is the pair $(-0, +0)$.

The standard Level 3 representation of a decorated interval datum $\boldsymbol{x}_{dx}$ is an ordered triple

$$\big(\texttt{inf}(x_{dx}),\ \texttt{sup}(x_{dx}),\ \texttt{decorationPart}(x_{dx})\big)$$

of two b64 datums and a decoration. For example, the only representative of $\mathrm{Empty}_{\texttt{trv}}$ is the triple $(+\infty, -\infty, \texttt{trv})$, and the only representative of NaI is the triple $(\mathrm{NaN}, \mathrm{NaN}, \texttt{ill})$.

Export and import of interchange formats normally occurs as a sequence of **octets** (bit strings of length 8, equivalently 8-bit bytes), e.g. in a file or a network packet.

At Level 4, interval objects are encoded as bit strings. We define an **octet-encoding** that maps the conceptual Level 3 representation into an octet sequence that comprises, in the order defined above, the interchange octet-encodings of the two `b64` datums, and, for decorated intervals, the decoration represented as an octet:

`ill`	00000000
`trv`	00000100
`def`	00001000
`dac`	00001100
`com`	00010000

NOTE—This encoding of decorations permits future refinements without disturbing the propagation order of the decorations.

The octet-encoding of `b64` datums is eight octets obtained from the 64 bits of the IEEE 754 interchange format: a sign bit, followed by 11 exponent bits that describe the exponent offset by a bias, and 52 bits that describe the significand (the least significant bit is last).

In Big-Endian octet-encoding, the first octet contains the sign bit and the 7 most-significant exponent bits. In Little-Endian octet-encoding, the first octet contains the 8 least-significant bits.

[Example. The Big-Endian interchange octet-encoding of $[-1, 3]_{\texttt{com}}$ are the concatenated octet sequences below

-1	10111111 11110000 00000000 00000000 00000000 00000000 00000000 00000000
3	01000000 00001000 00000000 00000000 00000000 00000000 00000000 00000000
`com`	00010000

The Little-Endian interchange octet-encoding of $[-1, 3]_{\texttt{com}}$ are the concatenated octet sequences below

-1	00000000 00000000 00000000 00000000 00000000 00000000 11110000 10111111
3	00000000 00000000 00000000 00000000 00000000 00000000 00001000 01000000
`com`	00010000

]

ANNEX A

Not required features of IEEE Std 1788™-2015 (informative)

This Annex lists the features of IEEE Std 1788™-2015 that are not required in IEEE P1788.1. The corresponding subclauses in IEEE Std 1788™-2015 are given in parenthesis.

The following operations required in the set-based flavor of IEEE Std 1788™-2015 are not required in IEEE P1788.1:

All reverse-mode elementary functions (10.5.4)
Two-output division (10.5.5)
`mulRevToPair`
Boolean functions of intervals (10.5.10)
`less`
`precedes`
`strictLess`
`strictPrecedes`
Reduction operations (12.2.12)
`sum`
`dot`
`sumSquare`
`sumAbs`
Exact text representation (13.4)
`intervalToExact`
`exactToInterval`