14. Level 3 description

14.1. Level 3 introduction. Level 3 is where Level 2 datums are represented, and operations on them described, in terms of more primitive entities and operations. How this is done is implementation-defined. Implementation may be by hardware, software, or a combination of the two.

Level 3 entities are here called *objects*; they represent Level 2 datums and may be referred to as *concrete*, while the datums are *abstract*. An implementation shall behave as if the relation between Levels 2 and 3 is as follows.

- The set of boolean values, the set of integer values, the set of strings §10.1, the set $\mathbb{D}$ of decorations §11.2, and the set of the states returned by `overlap` function §10.7.4, 12.13.4 are regarded as being the same at Level 3 as at Levels 1 and 2.
- The bare interval objects are organized into disjoint sets, *concrete bare interval types*, that are in one-to-one correspondence with the abstract bare interval types of Level 2. As at Level 2, a decorated interval is an ordered pair (bare interval, decoration), so this induces a one-to-one correspondence between the abstract and the concrete decorated interval types.
- The number objects are organized into disjoint sets, *concrete number formats*, that are in one-to-one correspondence with the abstract number formats of Level 2.

Thus intervals of a particular type exist in four forms: bare or decorated, and in either case abstract datums at Level 2 or concrete objects at Level 3. Similarly, numbers of a particular format exist in two forms, abstract or concrete.

In this document, the same name is normally used for an abstract type or format and its concrete counterpart. This convention, and the term "object", are not intended to constrain the names that an implementation gives to types or formats, nor the data structures it uses.
[*Example. The format* `decimal64` *might denote either the set of representations (in the sense of IEEE 754 §3.2) of decimal64 numbers, or the set of numbers thus represented. Then for instance, all representations in the cohort (IEEE 754 §3.5.1) of floating-point number* 0.1 *are different objects, but represent the same datum.*]

14.2. Representation. Individual datums of an abstract type or format are *represented* by individual objects of its concrete type or format. While the correspondence between abstract and concrete types or formats as a whole is one-to-one, that between datums and objects is not so. The property that defines a representation, for a given type or format, is:

> Each datum shall be represented by at least one object. Each object shall represent at most one datum. (35)

An object that represents a datum is called *valid*; one that does not is called *invalid*.

That is, representation is a *partial function* that is *onto* but usually *not one-to-one*, from the set of objects to the set of datums of a given type or format. The set of valid objects is the domain of this function.
[*Examples. Let* $\mathbb{F}$ *be a 754 format and let* $\mathbb{T}$ *the derived (bare) inf-sup type. Three possible representations are:*

- **inf-sup** *form. Any* $\mathbb{T}$*-interval* $\boldsymbol{x}$ *is represented at Level 3 by the object* $(\texttt{inf}(x), \texttt{sup}(x))$ *of two Level 2 numbers – members of* F*. All intervals have only one Level 3 representation because operations* `inf` *and* `sup` *are uniquely defined at Level 2 §12.12.9: 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 members of* $\mathbb{F}$*. A nonempty* $\mathbb{T}$*-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* $(\mathrm{NaN}, \mathrm{NaN})$*. Its valid objects are* $(\mathrm{NaN}, \mathrm{NaN})$*, together with all* (l, u) *such that* l, u *are not* NaN *and* $l \le u$*,* $l < +\infty$*,* $u > -\infty$*.*
- **neginf-sup-nan** *form. This is as the previous, except that the value of* l *is* $-\underline{x}$*. Its valid objects are* $(\mathrm{NaN}, \mathrm{NaN})$*, together with all* (l, u) *such that* l, u *are not* NaN *and* $0 \le l + u$*,* $l > -\infty$*,* $u > -\infty$*.*

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* Empty $= (\mathrm{NaN}, \mathrm{NaN})$.
]

14.3. 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. That is, let $y = \varphi(x_1, x_2, \ldots)$ be a Level 2 operation instance whose inputs and output are any mix of number, interval, decoration, string or boolean datums, and let objects $x'_1, x'_2, \ldots$ represent $x_1, x_2, \ldots$, respectively. Then $y' = \varphi(x'_1, x'_2, \ldots)$ shall be defined and be a representative of y.

Since for each Level 2 operation, the result is defined for arbitrary input datums, it follows that each Level 3 library operation has a unique result, up to representation, for arbitrary *valid* input objects. That is, if one chooses different representatives x'_i for the x_i, the result y' may be different but is still a representative of y. The result, when some inputs are invalid, is implementation-defined. An implementation shall provide means for the user to specify that an exception `InvalidOperand` to be signaled when this occurs.

To promote reproducibility (Annex B), an implementation should provide a computational mode where, at least for library operations with numeric output, the representative of the output is independent of the representatives of the inputs. That is, in the notation above, y' is not changed by changing the x'_i. [*Example. Let* $\mathbb{F}$ *be a 754 decimal format and* $\mathbb{T}$ *the derived inf-sup type. Suppose a nonempty* $\mathbb{T}$*-interval* $[l, u]$ *is represented at Level 3 as the pair of* $\mathbb{F}$*-numbers* (l, u)*. Let* f *be the expression*

$$\mathtt{sameQuantum}(\mathtt{inf}(\boldsymbol{x}), 0.3)$$

(see 754 §5.7.3), and consider $\boldsymbol{x} = [0.3, inf]$ *with the two Level 3 representations* $\boldsymbol{x}' = (0.3, +inf)$ *and* $\boldsymbol{x}'' = (0.30, +inf)$*. The notations* 0.3 *and* 0.30 *stand for 754 Level 3 objects* $(0, -1, 3)$ *and* $(0, -2, 30)$ *with the same value* $3 \times 10^{-1} = 30 \times 10^{-2}$*. Then* $\boldsymbol{x}' = \boldsymbol{x}''$ *in the Level 2 sense, but a naive implementation gives*

$$\begin{aligned} f(\boldsymbol{x}') &= \mathtt{sameQuantum}(\mathtt{inf}(\boldsymbol{x}'), 0.3) &&= \mathtt{sameQuantum}(0.3, 0.3) &&= \mathtt{true};\\ f(\boldsymbol{x}'') &= \mathtt{sameQuantum}(\mathtt{inf}(\boldsymbol{x}''), 0.3) &&= \mathtt{sameQuantum}(0.30, 0.3) &&= \mathtt{false}. \end{aligned}$$

The standard does not say which of these two results is "correct". But since they differ, the equality principle is violated and such an implementation is non-reproducible. One way to make it reproducible is to canonicalize all operations with numeric output, to ensure that for instance to return the member of the result's cohort with minimal decimal exponent.]

14.4. Interchange representations. The purpose of interchange representations is to allow the loss-free exchange of Level 2 interval data between 754-conforming implementations. This is done by imposing a standard Level 3 representation using Level 2 number datums and delegating choice of interchange format of number datums to the IEEE 754 standard.

Let $\boldsymbol{x}$ be a datum of the bare interval type T that is **inf-sup** $\mathbb{F}$ derived from a supported (754) format F. Its standard Level 3 representative is an ordered pair $(\mathtt{inf}(x), \mathtt{sup}(x))$ of two Level 2 F-numbers as defined in §12.12.9. For example, the only representative of Empty is the pair $(+\infty, -\infty)$ and the only representative of $[0, 0]$ is the pair $(-0, +0)$.

Let $\boldsymbol{x}_{dx}$ be a datum of the decorated interval type DT derived from the T. Its standard Level 3 representative is an ordered triple $(\mathtt{inf}(x_{dx}), \mathtt{sup}(x_{dx}), \mathtt{decorationPart}(x_{dx}))$ of two Level 2 F-datums and of a decoration. For example, the only representative of $\mathrm{Empty}_{\mathtt{trv}}$ is the triple $(+\infty, -\infty, \mathtt{trv})$ and the only representative of NaI is the triple $(\mathrm{NaN}, \mathrm{NaN}, \mathtt{ill})$.

Interchange representation of an interval datum comprises the interchange representations of fields of the ordered pair/triple above, in the order given. Interchange representation of the decoration field is an integer according with the table

Decoration	`ill`	`trv`	`def`	`dac`	`com`
Representation	0	1	2	3	4

Interchange representations of fields are not completely specified by this standard. Choices that are implementation-defined, and that an implementation shall document, include:

– choice of wider 754 interchange format if F is not 754 interchange format (754 §3.6) itself.

– choice of densely packed decimal (DPD) or binary integer decimal (BID) significand encoding for decimal 754 interchange formats.
– choice of size in bytes of integer decoration representation (one byte is recommended).

Issues of endianness, which affect how interchange representations map to sequences of bytes at Level 4, are outside the scope of the standard.

[*Note. The above rules imply an interval has a unique interchange representation if it is not* NaI *and in a binary format, but not generally otherwise. The reason for the rules is that the sign of a zero bound cannot convey any information relevant to intervals; but an implementation may potentially use cohort information, or a* NaN *payload.*]

A 754-conforming implementation shall provide an interchange representation for each supported 754 interval type. Interchange representations for non-754 interval types, and on non-754 systems, are implementation-defined. If an implementation provides other decoration attributes besides the standard ones, then how it maps them to an interchange representation is implementation-defined.