TEXT AND RATIONALE FOR MOTION 6: MULTI-FORMAT SUPPORT
POSITION PAPER P1788/PP013
VERSION 04, August 11, 2009

JOHN PRYCE, TECHNICAL EDITOR

1. Motion title and authors

Motion: P1788/M0006.04_Level_2_Multi-format
Proposer: John Pryce.
Seconder: Dan Zuras

2. Motion text

P1788 shall support multiple-format interval arithmetic. Specifically:

2.1. The terms **interval format**, **interval datum** etc., shall have the meanings given in the Definitions and Rationale.

2.2. An implementation shall specify which interval formats it supports (it is permitted to support only one). Support consists of the following two items.

(i) For each supported interval format, operations (as P1788 will in due course specify) shall be provided, to at least single-radix single-format (SRSF) level—see Section 3.7.
(ii) Conversions shall be provided between any two supported interval formats — see Section 3.6.4.

2.3. An implementation shall be called **754-conforming** (for a particular set of formats) if: (i) its underlying system is 754-conforming (Definition 3.3.1) for this set of formats, and the implementation supports the corresponding interval format to at least SRSF level; or (ii) it is functionally indistinguishable from case (i).

[Note. The reason for case (ii) is that it seems feasible to code an efficient 754-conforming interval system using only a subset of 754 floating-point features. If so, a floating-point system that behaves differently from 754 outside this subset can be used. For example, a system that has only one kind of zero.]

2.4. A 754-conforming implementation shall provide single-radix multiple-format (SRMF) interval support to the same extent that the underlying floating-point provides *formatOf* operations, see Section 3.7. It may support interval formats outside the five basic 754 formats, such as extended or extendable, see 754§3.7.

2.5. An implementation shall document:

(i) Which interval formats are supported.
(ii) Which interval versions of elementary functions are provided in a given format.
(iii) For each such version:
  (a) a sharpness measure, as P1788 will decide (e.g. Vienna's "tight", "accurate", "valid");
  (b) the level of mixed-format support, SRSF or SRMF; and just which operand interval formats are supported in the SRMF case.

3. Rationale

This divides into Foundations; Terminology and notation; Definitions; Overall Aims; Level 1 description; Level 2 description; and Support levels for interval elementary functions.

3.1. **Foundations.** Motion 2 and the associated position paper PP008, defining a levels structure as a guideline for the standard.

Motion 3, which defines P1788's number system and set of intervals.

3.2. **Terminology and Notation.** 754§X denotes section X in the standard IEEE754-2008. Vienna§X stands for section X in the final version of the Vienna proposal, 19 December 2008. "Einarsson–Kulisch" refers to the motion under discussion as motion 5 at the time of writing (June 09), and its supporting position paper.

Motion 3 decided that the P1788 intervals comprise precisely the elements of the set of all closed and connected subsets of the reals. Discussions with Kulisch, Neumaier and others resulted in the notation $\overline{\mathbb{IR}}$ for this set, and $\mathbb{IR}$ for the "classical" set of *nonempty, bounded* closed intervals. These notations are used here and in the latest revision of Einarsson–Kulisch, and hopefully will be acceptable to the group for use in the standard text. [Note that $\overline{\mathbb{IR}}$ is the lattice-theoretic completion of $\mathbb{IR}$, with containment as partial order.]

Usually, point values are denoted by normal-weight symbols $x, y, \ldots$ and intervals by bold symbols $\boldsymbol{x}, \boldsymbol{y}, \ldots$.

3.3. **Definitions.** [Note. These are intended to be rewritten into a standard set of definitions, abbreviations and acronyms of this standard. I have put them into alphabetical rather than logical order (as in 754§2) for ease of looking up. Please bear with the fact that some of them are currently just forward references.]

3.3.1. **754-conforming system** (**754 system** for short). A programming environment that provides floating-point arithmetic conforming to IEEE754-2008.
[Note. A hardware processor (e.g. the Cell processor) may not be 754-conforming in itself, but a 754-conforming system may be built on it with software assistance.]

3.3.2. **an-format** (**abstract number format**) and an-format **associated to** a concrete number format. See Section 3.6.1.

3.3.3. **ai-format** (**abstract interval format**) and ai-format **associated to** a concrete interval format. See Section 3.6.2.

3.3.4. **basic 754 format**. One of the five 754 floating-point formats binary32, binary64, binary128, decimal64, decimal128.

3.3.5. **basic operation**. One of the five elementary functions $+ \ - \ * \ /$ and $\sqrt{\ }$.

3.3.6. **cn-format, ci-format** (**concrete point/interval format**). A point/interval format associated with a particular representation. See Section 3.6.1 and 3.6.2.

3.3.7. **hull**. The (interval) hull of a subset of $\mathbb{R}$ is the tightest P1788 interval containing that subset.

3.3.8. **implementation**. When used without qualification, means an implementation of this standard, P1788.

3.3.9. **infsup**. Describes a representation of an interval based on its lower and upper bounds.

3.3.10. **interval datum**, **$\mathbb{F}$-interval datum**. See Section 3.6.2.

3.3.11. **interval elementary function**. An interval version of a point elementary function, that is provided by an implementation. The set of these is the implementation's **interval elementary function library** (**interval library** for short). These terms may be qualified by a format, e.g. "binary64 interval library".

3.3.12. **interval extension** of a point function. See Section 3.5.5.

3.3.13. **interval format**. See Definition 3.3.3 and Definition 3.3.6.

When used without qualification, **interval format** means an infsup interval format—by contrast with, say, a midrad format.

3.3.14. **interval function**, **interval mapping**. A function from intervals to intervals is called an **interval mapping**. If it is an interval extension of a point function, it is also called an **interval function**. See Section 3.5.5.

3.3.15. **interval version** of a point function. The same as interval extension; but often used with an indication of its operand and destination formats, as in "binary64 SRMF interval version".

3.3.16. **midrad**. Describes a representation of an interval based on its midpoint and radius.

3.3.17. **mixed-format** interval arithmetic operation. One where the formats of the operand interval(s) and the destination interval may not all be the same.

[Note. Mixed-format does not mean that the lower and upper bounds of an individual interval (represented in infsup form) can have different floating-point formats. The theory, as phrased here, precludes this.]

3.3.18. **multiple-format**. Multiple-format interval arithmetic means supporting arithmetic in more than one interval format and conversions between these formats.

[Note. SRSF, SRMF and MRMF interval support are all multiple-format (if more than one interval format is supported); SRMF and MRMF, but not SRSF, are mixed-format.]

3.3.19. **point elementary function**. A point function, in the sense of Section 3.5.4, that is provided by an implementation. The set of these is the implementation's **point elementary function library** (**point library** for short). These terms may be qualified by a format, e.g. "binary64 point library". The arithmetic operations $+ \; - \; * \; /$ count as elementary functions.

3.3.20. **sharpness measure**. A way to describe the quality of an interval version of a function: for instance the "tight", "accurate", "valid" scheme in Vienna§3.2.

3.3.21. **support**. An implementation **supports** an abstract interval format $\widetilde{\mathbb{IF}}$ if it provides a concrete interval format that represents $\widetilde{\mathbb{IF}}$. See Section 3.6.1 and 3.6.2.

A function f is **interval-supported** in an interval format $\widetilde{\mathbb{IF}}$ if there is an interval version of f whose destination format is $\widetilde{\mathbb{IF}}$. Levels of support (SRSF, SRMF and MRMF) are described in Section 3.7.

3.4. **Overall Aims.** Ramon Moore's Fundamental Theorem of Interval Arithmetic (FTIA) is central to interval computation. Roughly, it says that if E is an explicit real expression defining a real function $f(x_1, \ldots, x_n)$, then evaluating E "in interval mode" over any interval inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is guaranteed to give an enclosure of the range of f over those inputs.

This motion has two aims, linked to each other and to the FTIA.

Aim 1 is to define a framework for finite precision interval arithmetic in multiple formats. In this framework the FTIA is easily proved—indeed almost obvious. The finite precision interval operations are defined to operate on a set of abstract objects that happens—except for Not an Interval (NaI)—to be a finite subset of the infinite set $\overline{\mathbb{IR}}$. The definition of operations is independent of any representation that may be chosen for intervals.

The framework should permit mixing intervals of different formats in arithmetic expressions, as well as potential compile-time "exact denotations" of intervals such as $(\pi + [-0.1, 0.1])$. However, this motion does not commit P1788 to any stance on such issues.

[Note: To simplify wording I am assuming P1788 defines an NaI object. Personally I support NaI; editorially I have no preference; and NaI is peripheral to the argument.]

This is exactly analogous to how 754 defines the finite precision point operations, on a set of abstract objects that happens—except for –0, +0 and NaN—to be a finite subset of the extended real numbers.

The abstract objects and operations form interval level 2. Comparing with the Einarsson-Kulisch position paper, level 2 corresponds to its initial "declarative" definitions of operations, while the later "procedural" definitions belong to level 3. To see the utility of level 2, ask the question "How to prove the FTIA, starting from the procedural definitions?" Answer: you cannot, except by passing through a stage equivalent to the level 2 abstract objects and operations.

Aim 2 is to show that this abstract multiple-format arithmetic is, for the most frequently used operations, easy to implement efficiently on a 754 system, by exploiting two features of the 754 standard:

1. Every floating-point computational operation (754§2.1.11) has a defined destination format.
2. The basic operations $+ \ - \ * \ /$ and $\sqrt{\ }$ are *formatOf* operations (754§5.1 and §5.4.1), which means that the correctly rounded result is produced for any operands *of the same radix* as the destination format.

This makes mixed-format interval arithmetic for the basic operations, between formats of the same radix, only slightly more work to implement sharply (i.e., with tightest possible enclosures for each individual operation) than is single-format interval arithmetic. Mixed-radix is less easy, as there is no one best way to do the needed radix conversions. In my view it should not be considered.

In summary, Level 2, as defined here, has a precise relation to the mathematical level 1, that is easily grasped and easily reasoned about. It also offers—to turn this into reality is the job of implementers—a precise and easily grasped relation to the representation/algorithmic level 3. Hence it provides a layer between mathematics and implementation that will be crucial for proving the correctness of this standard, and of programs built upon it.

3.5. **Level 1 description.** This section recapitulates matters at the mathematical level that have, I believe, already been decided.

I believe nothing in this motion and rationale hinders the implementation of various forms of non-standard intervals—Kahan, modal, etc.—as discussed at the end of Vienna§1.2. Their theory is incompatible with certain *representations*, e.g., with a finite precision midrad (midpoint-radius) representation of intervals, though there is nothing to stop an algorithm using this internally.

3.5.1. $\mathbb{R}$ is the set of reals. $\mathbb{R}^*$ is the set of extended reals, namely $\mathbb{R} \cup \{-\infty, +\infty\}$. Using the terminology of 754 (754§2.1.25 and elsewhere), any member of $\mathbb{R}^*$ is called a number: it is a **finite number** if it is in $\mathbb{R}$, else an **infinite number**.

3.5.2. Following Motion 3, the set of **textbook intervals** (Vienna §1.2) in $\mathbb{R}$, denoted $\overline{\mathbb{IR}}$, comprises the empty set $\emptyset$ together with all closed nonempty intervals of real numbers

$$\boldsymbol{x} = [\underline{x}, \overline{x}] := \{\, x \in \mathbb{R} \mid \underline{x} \le x \le \overline{x} \,\},$$

where $-\infty \le \underline{x} \le \overline{x} \le +\infty$.

The above definition implies $-\infty$ and $+\infty$ are never members of an interval. Consistent with the traditional notation $(\underline{x}, \overline{x}] := \{\, x \in \mathbb{R} \mid \underline{x} < x \le \overline{x} \,\}$, $[\underline{x}, \overline{x}) := \{\, x \in \mathbb{R} \mid \underline{x} \le x < \overline{x} \,\}$, and $(\underline{x}, \overline{x}) := \{\, x \in \mathbb{R} \mid \underline{x} < x < \overline{x} \,\}$, the round bracket notation for closed intervals with an infinite end point can be used; e.g. $[2, +\infty)$ is the same as $[2, +\infty]$.

The requirement that $\boldsymbol{x}$ be nonempty implies $\underline{x}$ cannot be $+\infty$, and $\overline{x}$ cannot be $-\infty$; we treat $[-\infty, -\infty]$ and $[+\infty, +\infty]$ as having no meaning (rather than being empty).

3.5.3. The (interval) **hull** of an arbitrary subset $\boldsymbol{s}$ of $\mathbb{R}$, written hull($\boldsymbol{s}$), is the tightest member of $\overline{\mathbb{IR}}$ that contains $\boldsymbol{s}$. (The **tightest** set with a given property is the intersection of all other sets having that property, provided the intersection itself has this property.)

3.5.4. A **point function** is a (possibly partial) multivariate real function: that is, a mapping f from a subset D of $\mathbb{R}^n$ to $\mathbb{R}^m$ for some integers $n \ge 0$, $m > 0$. (When $n = 0$ and $m = 1$, we have a **named real constant**.) When not otherwise specified, a scalar function is assumed, i.e. $m = 1$. If $m > 1$, the function is called a vector function. D is the domain of f, also written D_f. To specify n, call f an n**-variable point function**, or denote f as

$$f(x_1, \ldots, x_n).$$

The **range** (often called **exact range** in the literature) of f over an arbitrary subset $\boldsymbol{s}$ of $\mathbb{R}^n$ is the set

$$\text{range}(f; \boldsymbol{s}) = \{\, f(x) \mid x \in \boldsymbol{s} \text{ and } x \in D_f \,\}.$$

Equivalently, for the case where $\boldsymbol{f}$ has separate arguments $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$, each being a subset of $\mathbb{R}$, the range is written as

$$\text{range}(f; \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$$

This is an alternative notation for the case where $\boldsymbol{s}$ is the cartesian product of the $\boldsymbol{s}_i$.

Notes.

1. Here, f is a mapping, not an expression.
2. For instance range($\sqrt{\ }$; $[-1, 1]$) = $[0, 1]$. That is, we follow the convention, usual in mathematics, that when evaluating over sets, points outside D_f are simply ignored. The Vienna proposal, and Einarsson–Kulisch, make the same definition.
3. This motion does not commit P1788 to any specific means of signalling evaluation outside the domain of a function, such as a flag or exception.
4. For the 754 policy on evaluating point functions outside the domain, see 754§9.1.1.

3.5.5. Unless otherwise specified, an **interval mapping** is a mapping $\boldsymbol{f}$ from $\overline{\mathbb{IR}}^n$ to $\overline{\mathbb{IR}}^m$ for some $n \geq 0$, $m > 0$. To specify n, call $\boldsymbol{f}$ an "n-variable interval mapping", or denote $\boldsymbol{f}$ as $\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. As with point functions, $m = 1$ is assumed unless said otherwise.

An interval mapping is called an **interval function** if it is an interval version of some point function, as defined next. Examples of interval mappings that are not interval functions are the intersection and interval-union operations, $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x} \cap \boldsymbol{y}$ and $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \mathrm{hull}(\boldsymbol{x} \cup \boldsymbol{y})$.

Given an n-variable point function f, an **interval extension** of f, also called an **interval version** of f, is any interval mapping $\boldsymbol{f}$ such that

$$\boldsymbol{f}(\boldsymbol{s}) \supseteq \mathrm{range}(f; \boldsymbol{s}) \text{ for any subset } \boldsymbol{s} \text{ of } \mathbb{R}^n.$$

The **sharp interval extension** of f is defined by

$$\boldsymbol{f}(\boldsymbol{s}) = \mathrm{hull}(\mathrm{range}(f; \boldsymbol{s})) \text{ for any subset } \boldsymbol{s} \text{ of } \mathbb{R}^n.$$

Equivalently, using multiple-argument notation for $\boldsymbol{f}$, an interval extension satisfies

$$\boldsymbol{f}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) \supseteq \mathrm{range}(f; \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n),$$

and the sharp interval extension satisfies

$$\boldsymbol{f}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) = \mathrm{hull}(\mathrm{range}(f; \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)).$$

When f is a binary operator $\bullet$ written in infix notation, this gives the usual definition of its (sharp) interval extension as

$$\boldsymbol{x} \bullet \boldsymbol{y} = \mathrm{hull}(\{\, x \bullet y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y}, \text{ and } x \bullet y \text{ is defined} \,\}).$$

Notes.

1. Example. With these definitions, $\boldsymbol{x} * \{0\} = \{0\}$ for any nonempty interval $\boldsymbol{x}$, and $\boldsymbol{x}/0 = \emptyset$, for any interval $\boldsymbol{x}$.
2. All interval functions used here are automatically defined for all arguments—e.g. for the sharp extension of "point square root",

$$\sqrt{[-1, 4]} = [0, 2], \quad \sqrt{[-2, -1]} = \emptyset.$$

3.6. **Level 2 description.**

3.6.1. An **abstract number format** (**an-format**) $\mathbb{F}$ is a finite subset of $\mathbb{R}^*$ containing $-\infty$ and $+\infty$.

A format in the 754 sense (a **concrete number format**, **cn-format**, such as binary64) shall be identified with the an-format comprising those extended-real numbers that are exactly representable in that format, where –0 and +0 both represent 0. This is the an-format **associated to** the concrete format.

Notes.

1. In view of this definition, 754's –0 and +0 are considered identical. Also, in 754 decimal formats, numbers in the same cohort are considered identical.
2. For examples, we use the abbreviations b64 to mean 754's 64-bit binary format, d64 for 64-bit decimal, and so on.

3.6.2. An **$\mathbb{F}$-interval**, for some an-format $\mathbb{F}$, is either the empty set, or a textbook interval whose endpoints are in $\mathbb{F}$. When it is necessary to distinguish, it is called an **infsup** (infimum-supremum) $\mathbb{F}$-interval by contrast to, say, an interval represented in a midrad (midpoint-radius) form. The set of all $\mathbb{F}$-intervals is denoted by $\overline{\mathbb{IF}}$.

An **$\mathbb{F}$-interval datum**, following the definition of floating-point datum in 754§3.2, is either

- an $\mathbb{F}$-interval; or
- the abstract object NaI, "Not an Interval".

The **abstract interval format** (**ai-format**) associated with $\mathbb{F}$ means the set of $\mathbb{F}$-interval datums. It is denoted by $\widetilde{\mathbb{IF}}$. Thus

$$\widetilde{\mathbb{IF}} = \overline{\mathbb{IF}} \cup \{\text{NaI}\}.$$

Note. At this level we assume that there is at most one NaI for all an-formats; at the representation level this will probably not be the case.

My view of the purpose of NaI is that it behaves as NaN (mostly) does in floating-point computation: if any operand to an operation is NaI, the result is NaI. Not everyone agrees with this definition, but this controversy is peripheral to multiple-format issues, so (without prejudice to what may be decided) the rest of this document ignores the case where some operand to an operation is NaI.

1. Clearly an an-format uniquely determines an ai-format (also vice versa), and a 754 concrete format uniquely determines an an-format.
2. An example of a possibly useful an-format that is not associated to a concrete format (but is derived from one), is an "underflows are flushed to zero" interval system. Say the concrete format is binary64. Then the nonempty intervals are those whose endpoints belong to $\mathbb{F}$, which is defined to consist of all binary64 numbers that are not subnormal. Such a system may give speed advantages on some architectures.
3. The above definitions imply some form of infsup, that is (lower bound, upper bound) representation at level 3. Variants of this exist, see for instance Vienna§1.6.
4. It is not just wrong, but actually meaningless at this level, to speak of an interval that has NaN as an endpoint or has its lower bound greater than its upper bound.

A **concrete interval format** or **ci-format**) is a surjective mapping from a set C of instances of a data structure (typically, a pair of floating-point datums) to an associated ai-format. There may be several ci-formats with the same data structure and the same ai-format, e.g., the `[xlo,xhi]` and `[-xlo,xhi]` representations of an interval.

3.6.3. The (interval) $\mathbb{F}$-hull of an arbitrary subset $\boldsymbol{s}$ of $\mathbb{R}$, written $\text{hull}_{\mathbb{F}}(\boldsymbol{s})$, is the tightest $\mathbb{F}$-interval that contains $\boldsymbol{s}$.

Notes.

1. The set $\text{hull}_{\mathbb{F}}(\boldsymbol{s})$ always exists, because $\mathbb{F}$ is finite and contains $\pm\infty$.
2. Always, $\text{hull}_{\mathbb{F}}(\boldsymbol{s})$ contains $\text{hull}(\boldsymbol{s})$. If an-format $\mathbb{G}$ (as a subset of $\mathbb{R}^*$) contains an-format $\mathbb{F}$ —equivalently in the 754 case, if format $\mathbb{G}$ is wider than $\mathbb{F}$ in the sense of 754§2.1.36—then $\text{hull}_{\mathbb{F}}(\boldsymbol{s})$ contains $\text{hull}_{\mathbb{G}}(\boldsymbol{s})$.

3.6.4. Interval format conversion. If $\mathbb{F}$ is an an-format, the **conversion** to format $\mathbb{F}$ shall mean the operation that maps an interval $\boldsymbol{x}$ of any other supported format to its $\mathbb{F}$-hull,

$$\boldsymbol{y} = \text{hull}_{\mathbb{F}}(\boldsymbol{x}).$$

On 754 systems, this interval operation can in all cases (whether $\boldsymbol{x}$ has the same radix as $\mathbb{F}$ or not) be implemented in terms of one of the floating-point operations *formatOf*-**convertFormat** defined in 754§5.4.2, with the appropriate outward rounding.

3.7. **Support levels for interval elementary functions.** The Fundamental Theorem of Interval Arithmetic (FTIA) relies on each point elementary function e in a real expression being replaced by an interval version $\boldsymbol{e}$. Mathematically, $\boldsymbol{e}$ can be an arbitrary interval extension of e, and its arguments and result are not limited by any concrete interval format.

Practically, a level 2 **interval version** must be implemented at level 3 in terms of concrete formats such as binary64. Typically, $\boldsymbol{e}$ is coded to deliver a result of a *specified* ci-format, from operands of a *limited number* of ci-formats. What should multiple-format support mean? For a given supported interval format $\mathbb{F}$ and an elementary function e, three levels of support are suggested by the design of the 754 standard. In all of them, conversions as in Section 3.6.4 are provided between all supported interval formats.

1. Single-radix, single-format (SRSF). In SRSF support, e has an interval version $\boldsymbol{e}$ that takes $\mathbb{F}$-interval operand(s) and gives an $\mathbb{F}$-interval result. Thus explicit format conversion is needed for any interval operand of a different format from $\mathbb{F}$.
2. Single-radix, mixed-format (SRMF). In SRMF support, e has an interval version $\boldsymbol{e}$ that takes operand(s) of any supported interval format of the same radix as $\mathbb{F}$, and gives an $\mathbb{F}$- interval result. Thus explicit format conversion is needed for any operand whose interval format has a different radix from $\mathbb{F}$.
3. Mixed-radix, mixed-format (MRMF). In MRMF support, e has an interval version $\boldsymbol{e}$ that takes operand(s) of any supported interval format, and gives an $\mathbb{F}$-interval result. Thus no explicit format conversion is required for any operand.

Clearly SRMF includes SRSF (SRMF support provides SRSF in particular); and MRMF includes SRMF. I believe (this is my personal view):

(a) A function being "interval-supported" should, in all cases, mean at least SRSF support.
(b) Since SRMF support is the interval equivalent of 754's *formatOf* operations, it is appropriate on a 754 system for the *formatOf* operations of 754§5.4.1, namely the basic operations (Definition 3.3.5) and `fusedMultiplyAdd`. (Also, `convertFromInt` is a *formatOf* operation, but it does not seem relevant for intervals.)
(c) Mandatory MRMF support is less appropriate, but P1788 may make recommendations on how it is done.

My reason for (b) is as follows. For a *formatOf* floating-point operation, whatever the input and output formats, provided they have the same radix, the correctly rounded result is produced in all cases. E.g. when adding two b64's, whether the destination format $\mathbb{F}$ is b32, b64 or b128, and whatever the rounding direction, the $\mathbb{F}$-number nearest the exact result in the relevant direction will be delivered. The *formatOf* concept eliminates the risk of "double rounding" error in mixed-format operations.

There are various algorithms for the basic interval operations—multiply and divide especially have variants optimised for different environments—but all those that I know of can exploit the *formatOf* feature. That is, all can be written, in terms of the point operations, so that arbitrary mixed formats of the same radix can be handled by essentially the same code, while remaining sharp, that is optimally tight, at the level of a single interval operation.

Therefore I believe the effort in implementing SRMF interval support of the basic operations, across all 754 formats supported by a system, is negligibly greater than that of implementing SRSF support for those operations in just one format such as binary64.

Finally, issues of accuracy (tightness) of interval elementary functions are largely independent of the SRSF/SRMF/MRMF issue and should be decided separately. Also, in mixed-format interval expressions (rather than individual operations) the destination format of each individual operation must be decided. The rules for this are language-dependent, though it is likely that P1788 will make recommendations and/or requirements about them, analogous to those for floating-point arithmetic in 754§10.