

IMPLICIT AND EXPLICIT INTERVAL FORMATS

VERSION 4

JOHN PRYCE

1. INTRODUCTION

This is a revision of the position paper I circulated in mid July, rewritten to propose a motion.

1.1. Background. There has been disagreement in the P1788 group for some time about the proper status of interval formats such as mid-rad, that are not based on storing the lower and upper bounds. There has also been criticism of the current draft text as too prescriptive: why go beyond a minimal specification that ensures the Fundamental Theorem of Interval Arithmetic (FTIA) holds? Personally I oppose the latter view: in particular I think our standard is obligated to demand more from implementations with underlying 754 arithmetic, to guarantee users the extra sharpness or efficiency such systems can provide. As for the former, I am sympathetic to both the “keep mid-rad out” and the “bring mid-rad in” views, so I think that a conceptual framework that includes mid-rad deserves to be put before the group for a vote.

There were discussions over several weeks in June–July 2010, mainly between Corliss, Hayes, Kearfott, Keil, Pryce and Zuras. These have produced a revised approach to the standard. I hereby propose it as a motion.

The basic change is allow an interval datatype—the new level 2 name for an interval format—to be either “explicit” (e.g. inf-sup datatypes) or “implicit” (e.g. mid-rad ones). I believe this gives a structure where

All interval datatypes are equal

in a way that satisfies the proponents of mid-rad representation. It may be that “some are more equal than others”, but this is due to intrinsic differences between implicit and explicit types.

This motion is about approving the general explicit/implicit concept. However, *conformance-defining* passages—“Master, what shall I do to be saved?”—are the interface between the standard and its implementations. Of these in the current draft standard (DS) the key one is in DS§3.2. I think the revision of this should be part of the motion, to give meat for the group to dig its teeth into during the discussion period. It seemed to me to need only small changes, and my suggested rewrite is in 3.4. (I have not corrected the missing cross references ‘??’, which are to be interpreted as in the current DS.)

After the statement of the motion in Section 2, Section 3 describes the new definitions and reasons for them. Section 4 argues the case for making the hull operation part of an implicit idatatype’s definition. Section 5 raises points for discussion of possible consequential changes to the standard. *The motion is not about details of such changes.* The devil is in the detail, and it is possible that some horrible flaw in the whole concept may appear. But I think “implicit” is sufficiently loosely coupled to existing concepts to make this very unlikely.

1.2. New terms. Interpret references in this motion to the current draft standard text, version 02.2, as if, throughout the latter, “interval format” (iformat) has been renamed to “interval datatype” (idatatype) and “concrete interval format” to “interval format” (iformat).

2. MOTION

1. The notion of an idatatype and arithmetic thereon shall be redefined as in 3.1, 3.2, and the standard shall permit both *implicit* and *explicit* idatypes.
2. The definition of conformance in DS§3.2 shall be replaced by the text in 3.4.

3. RATIONALE FOR THE EXPLICIT/IMPLICIT CONCEPT

Our standard, as decided by past motions and expressed in the current draft, provides:

- (a) a clear conceptual basis in the mathematical model and the four-level structure;
- (b) close (but optional) relation to the 754-2008 standard, important for tight enclosures and efficiency on 754-conforming systems;
- (c) built in multi-format support;
- (d) an efficient exception-handling mechanism;
- (e) the possibility to specify reproducible behaviour, of which more below.

However, being inherently based on an inf-sup representation, it does not satisfy those who ask that

Implementations based on other representations should be able to conform to the standard.

The revision here proposed meets that demand while not losing any of the listed features.

This motion is about *bare* intervals. The decoration scheme is unchanged; past motions have established it can provide a “Not an Interval” facility. Hence all level 2 interval datums represent level 1 mathematical intervals—there are no special symbols like NaI.

3.1. New definitions.

Idatatype. The old definition only supported *inf-sup* idatatypes: those whose nonempty intervals comprise all $[\underline{x}, \bar{x}]$ where \underline{x} and \bar{x} are in a specific set \mathbb{F} , which enforces an inf-sup representation or equivalent. In the new definition, an idatatype comprises:

- a finite set $\overline{\mathbb{ID}}$ of mathematical intervals¹;
- a *hull operation* $\mathbf{y} = \text{hull}_{\overline{\mathbb{ID}}}(S)$, defined for any subset S of \mathbb{R} , such that \mathbf{y} is in $\overline{\mathbb{ID}}$ and contains S , and is *minimal* in the sense that if \mathbf{z} is in $\overline{\mathbb{ID}}$ and $S \subseteq \mathbf{z} \subseteq \mathbf{y}$, then $\mathbf{z} = \mathbf{y}$.

It follows from this definition that the whole line \mathbb{R} must always be a member of $\overline{\mathbb{ID}}$. Apart from this, $\overline{\mathbb{ID}}$ is in principle arbitrary.

Explicit and implicit. $\overline{\mathbb{ID}}$ is *explicit* if any subset S of \mathbb{R} has a unique tightest (smallest in the order defined by set inclusion) enclosing member of $\overline{\mathbb{ID}}$. Otherwise $\overline{\mathbb{ID}}$ is *implicit*.

[*Note. Clearly all inf-sup idatatypes are explicit. Other explicit idatatypes exist, for instance the set of symmetric intervals $\{[-b, b] \mid b \in B\}$ where B is some finite set of non-negative numbers including ∞ . It is easy to see that all idatatypes based on a nontrivial mid-rad representation are implicit.*]

An explicit $\overline{\mathbb{ID}}$ automatically has a *uniquely defined* hull operation, which thus doesn’t need to be specified: the hull of S is the tightest member of $\overline{\mathbb{ID}}$ enclosing S . For an implicit $\overline{\mathbb{ID}}$ the hull operation is not uniquely defined.

3.2. Arithmetic. Arithmetic on $\overline{\mathbb{ID}}$ is defined abstractly in terms of levels 1 and 2 with no reference to floating-point arithmetic. An $\overline{\mathbb{ID}}$ -mapping means a function that takes $\overline{\mathbb{ID}}$ intervals as input and gives $\overline{\mathbb{ID}}$ intervals as output. To make the Fundamental Theorem of Interval Arithmetic work we need enclosure so, if $f(x, y, \dots)$ is a point operation, then

- A *valid* $\overline{\mathbb{ID}}$ -version of f is any $\overline{\mathbb{ID}}$ -mapping $\mathbf{f}(x, y, \dots)$ that encloses the exact range, i.e. for any x, y, \dots in $\overline{\mathbb{ID}}$ the interval $\mathbf{r} = \mathbf{f}(x, y, \dots)$ belongs to $\overline{\mathbb{ID}}$ and satisfies

$$\mathbf{r} \supseteq \text{range}(f; x, y, \dots) =_{\text{def}} \{ f(x, y, \dots) \mid x \in x, y \in y, \dots \text{ and this } f \text{ value is defined} \}$$

for all intervals x, y, \dots in $\overline{\mathbb{ID}}$.

- The unique *tightest* $\overline{\mathbb{ID}}$ -version of f is that for which \mathbf{r} is always the hull of the exact range:

$$\mathbf{f}(x, y, \dots) = \text{hull}_{\overline{\mathbb{ID}}}(\text{range}(f; x, y, \dots)).$$

¹The bar on top means “including unbounded intervals”. I don’t much like it but it follows our convention from an earlier motion.

[Note. This follows the Vienna proposal’s usage of “valid” and “tightest”. For explicit, inf-sup-based idatatypes it is equivalent to the current definition.

We have not yet had a motion on required tightness of standard functions for inf-sup idatatypes, let alone general ones, so this motion is silent on the issue. The rules for implicit idatatypes will be less restrictive than for inf-sup ones—after all, trading tightness for speed is one of the reasons for using types such as mid-rad. But I argue below that tightest versions should be available for optional use, for as many idatatypes as possible.]

3.3. Some things that are unchanged. References DS§X are to clause X in draft standard version 02.2.

- $\overline{\mathbb{D}}$ is a set of things (interval datums) which viewed at level 2 have no internal structure, but viewed at level 1 have the internal structure of being sets of points; while their level 3 representations give them internal structure of another kind, e.g. as pairs of floating-point numbers.
- (DS§6.1) An *iformat* is a representation of an idatatype, namely a set $\overline{\mathbb{F}}$ whose members are called (level 3) interval objects, together with a map $\overline{\mathbb{F}} \rightarrow \overline{\mathbb{D}}$. If object X maps to datum \mathbf{x} , we say X represents \mathbf{x} . The map is surjective, i.e. every datum has a representation.
The representation is *lossless*—it means what it says. E.g., a mid-rad form with midpoint $m = 1$ and radius $r = 1\text{e-}300$ means precisely the mathematical interval $[1 - 10^{-300}, 1 + 10^{-300}]$.
- (Mainly DS§6.2,6.3,6.4) A class of 754-conforming idatatypes is singled out, whose more stringent rules exploit the 754-2008 standard to give, for instance, optimal tightness in mixed-idatatype arithmetic (the “formatOf” feature).

3.4. Proposed revised Conformance requirements (DS§3.2).

- Implementations shall be based on the mathematical model in Section ??.
- An implementation shall support at least one idatatype. Support consists of the following two items.
 - (a) For each supported idatatype, operations as specified in ?? shall be provided, to at least single-datatype (SD) level—see ??.
 - (b) Conversions as defined in ?? shall be provided between any two supported idatatypes.
- A conforming implementation shall be **754-conforming** (for a specified set of its idatatypes which need not be all those that it supports) if it satisfies these two requirements:
 - (a) Its underlying system either (i) is 754-conforming (Definition ??) for this set of idatatypes; or (ii) is functionally indistinguishable from case (i).
 - (b) It satisfies the extra requirements for 754-conformance that are stated at various places in the standard.

[Note. The reason for case (a)(ii) is that it seems feasible to code an efficient 754-conforming interval system using only a subset of 754 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.]
- A 754-conforming implementation shall support operations on its idatatypes to at least single-datatype (SD) level, and shall provide single-radix multiple-datatype (SRMD) support to the same extent that the underlying floating-point provides *formatOf* operations, see ??. It may support idatatypes outside the five basic 754 formats, such as extended or extendable, see 754§3.7.
- An implementation shall document:
 - (a) Which idatatypes are supported.
 - (b) Which interval versions of elementary functions are provided in a given idatatype.
 - (c) For each such version: (i) a sharpness measure as defined in ??; (ii) the level of mixed-idatatype support, SD or SRMD; and which operand idatatypes are supported in the SRMD case.

[Note. The implementer of an implicit idatatype is not required to document the level 3 representation, which may be proprietary.]

4. RATIONALE FOR DEFINED HULLS, AND REPRODUCIBILITY

4.1. The hull. The decision whether the hull operation is made part of an idatatype’s definition affects (a) inter-idatatype conversion, which is done by forming the hull; (b) the definition of “tightest” standard functions; (c) hence reproducibility.

[*Example.* Use the notation $m \pm r$ to mean the interval $[m-r, m+r]$ written in mid-rad form. Let \mathbb{ID} comprise all intervals $m \pm r$ where m and r belong to the set \mathbb{F} of 4-digit decimal floating point numbers, with some finite exponent range that is irrelevant here.

What is the conversion of the inf-sup interval $[1, 1.003]$ to mid-rad? If $\text{hull}_{\mathbb{ID}}$ is not part of the definition of \mathbb{ID} , one implementation can choose 1.001 ± 0.002000 , another can choose 1.002 ± 0.002000 , and both are right.]

Whether one approves of this non-uniqueness depends on one’s philosophy of the standard: should it specify minimum demands consistent with the FTIA, or should it tie things down more closely?

Personally I agree with Dan Zuras (754 chair for a number of years) that one of the main reasons why the 754 floating-point standard has been so successful is because it took the hard road of specifying things down to the last bit. The parallel LIA standard, which didn’t, has sunk with barely a trace.

So let’s tie things down. What, and how far? I think requiring mid-rad implementors to define an unambiguous, platform-independent hull operation is not too much to ask. Nate Hayes agrees (Section 5 item 7).


4.2. Reproducibility. I believe reproducibility, important for floating-point, is doubly important for interval computing. With default compiler options, running the same interval code on different platforms is not expected to produce the same results down to the last bit. But a user should be able to choose a mode that (presumably at the expense of speed) ensures identical results on all platforms, for code restricting itself to some subset of language features.






There is a counter-argument that reproducibility is *less* important for interval computing: if different platforms give different results, good, because they both enclose the true result. I accept that but am unmoved by it. As Dan Zuras (13 July 2010) says, why should I trust *either* result? For instance, what if a specialised interval computing chip has something like the famous Pentium bug? The bug will probably be far easier to find if I can run the chip in “reproducibility mode”.

The key to reproducibility is reproducible behaviour of interval standard functions. The only reasonable way to specify this is to require these functions to return “tightest” results for all arguments. The remarkable work of the French experts means it will soon be practical to compute results correct to the last bit in either rounding direction for all (point) standard functions, all arguments and all sensible number formats, with little loss of speed. For inf-sup idatatypes this makes “tightest”, hence reproducible, standard functions entirely practicable.

For general idatatypes I do not know how hard it is to achieve the same level of tightness (but see Nate Hayes’ view below). In general, I believe any steps that promote reproducibility will in the long term make systems programmers and users alike more confident that our interval systems are correct.

5. FOR FURTHER DISCUSSION

The following is an incomplete list of previous passed motions and their applicability to the new scheme, plus some other issues, and some of  *my opinions*.

1. Motion 8: decorations.  *Applies to all idatatypes.*
2. Motion 9: exact dot product.  *I suggest to make it mandatory for 754-conforming idatatypes, recommended for the rest.*
3. Motion 10: list of “required” and “recommended” operations.  *Applies to all, I suggest.*
4. Motion 16: inf-sup and mid-rad.  *The “available” / “supported” distinction is needed; the rest seems subsumed in the current motion.*
5. Motion 17: I/O.  *Applies to all idatatypes.*

6. Motion 18: tetrts. ⚠ Applies to all idatatypes.
7. What operations, if any, should be required to be “tightest” for *all* idatatypes? Nate Hayes, who has studied the issues carefully, says (private email 12 Aug 2010)
For implicit and explicit types with fixed precision, returning a “tightest” hull for the basic operations $+$, $-$, $*$, $/$, and sqrt , as well as all conversions, is probably a reasonable thing to expect...
8. Conversions. It is proposed (by Hayes and Zuras I think) that for each implicit $\overline{\mathbb{D}}$, conversions shall be provided between $\overline{\mathbb{D}}$ and at least one “available” explicit $\overline{\mathbb{D}}'$, by forming the appropriate hull.