P1788 – Standardization of Interval Arithmetic

Vincent LEFÈVRE

AriC, INRIA Grenoble – Rhône-Alpes / LIP, ENS-Lyon

GdT AriC, 2012-06-14

Outline

- Introduction
- The 4 Levels
- Exception Handling / Decorations
- Comparison Relations
- Number Formats
- Level 1 Constructors
- Numeric Functions of Intervals

Introduction

P1788 Scope: "This standard specifies basic *interval arithmetic* (IA) operations selecting and following one of the commonly used *mathematical interval models*. This standard supports the IEEE-754/2008 floating point types of practical use in interval computations. *Exception conditions* will be defined and standard handling of these conditions will be specified. Consistency with the model is tempered with practical considerations based on input from representatives of vendors and owners of existing systems. The standard provides a layer between the hardware and the programming language levels. It *does not mandate* that any operations be implemented in hardware. It *does not define any realization* of the basic operations as functions in a programming language."

Summary of the last 3 years of discussions...

Interval Arithmetic: The Mathematical Model

Goals: To standardize

- conventional interval arithmetic;
- what could be regarded as "exceptions" (more generally, additional information given to the user, like continuity over the inputs).

Interval Arithmetic: The Mathematical Model

Goals: To standardize

- conventional interval arithmetic;
- what could be regarded as "exceptions" (more generally, additional information given to the user, like continuity over the inputs).

Intervals: Closed real intervals $\overline{\mathbb{IR}}$ (including unbounded intervals, i.e. with bounds in $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, and the empty set). Non-empty intervals:

$$\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\} \quad \text{where} \quad \underline{x} \leq \overline{x},\ \underline{x} \neq +\infty,\ \overline{x} \neq -\infty.$$

Interval Arithmetic: The Mathematical Model

Goals: To standardize

- conventional interval arithmetic;
- what could be regarded as "exceptions" (more generally, additional information given to the user, like continuity over the inputs).

Intervals: Closed real intervals $\overline{\mathbb{IR}}$ (including unbounded intervals, i.e. with bounds in $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, and the empty set). Non-empty intervals:

$$\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\} \quad \text{where} \quad \underline{x} \leq \overline{x},\ \underline{x} \neq +\infty,\ \overline{x} \neq -\infty.$$

Containment property: For any point function $f : \mathbb{R}^n \to \mathbb{R}$ of domain $\mathrm{Domain}(f)$, an *interval extension* $\mathbf{f} : \overline{\mathbb{IR}}^n \to \overline{\mathbb{IR}}$ must satisfy

$$\forall x \in \mathbf{x} \cap \mathrm{Domain}(f),\ f(x) \in \mathbf{f}(\mathbf{x}), \quad \text{i.e.} \quad \mathbf{f}(\mathbf{x}) \supseteq \mathrm{Range}(f|\mathbf{x}).$$

$\to$ An interval version of an arithmetic expression satisfies this property (FTIA-1).

Interval Arithmetic on the Machine

Representation (some intervals, like $\varnothing$, may need a special representation):

- by the bounds (*inf-sup*), best for generic intervals;
- by the midpoint m and the radius r (*mid-rad*), good only when r is "small" (approximation and error): $[\underline{x}, \overline{x}] = [m - r, m + r]$;
  Example: midpoint in arbitrary precision and radius in small precision.
- by a triple $\langle x_0, \underline{e}, \overline{e} \rangle$ (*triplex*): $[\underline{x}, \overline{x}] = x_0 + [\underline{e}, \overline{e}] = [x_0 + \underline{e}, x_0 + \overline{e}]$.

Interval Arithmetic on the Machine

Representation (some intervals, like $\varnothing$, may need a special representation):

- by the bounds (*inf-sup*), best for generic intervals;
- by the midpoint m and the radius r (*mid-rad*), good only when r is "small" (approximation and error): $[\underline{x}, \overline{x}] = [m - r, m + r]$;
  Example: midpoint in arbitrary precision and radius in small precision.
- by a triple $\langle x_0, \underline{e}, \overline{e} \rangle$ (*triplex*): $[\underline{x}, \overline{x}] = x_0 + [\underline{e}, \overline{e}] = [x_0 + \underline{e}, x_0 + \overline{e}]$.

Rounding:

- Containment property must be preserved for the library functions (outward rounding). $\rightarrow$ FTIA-1 still satisfied.
- Accuracy modes...

Interval Arithmetic on the Machine

Representation (some intervals, like $\varnothing$, may need a special representation):

- by the bounds (*inf-sup*), best for generic intervals;
- by the midpoint m and the radius r (*mid-rad*), good only when r is "small" (approximation and error): $[\underline{x}, \overline{x}] = [m - r, m + r]$;
  Example: midpoint in arbitrary precision and radius in small precision.
- by a triple $\langle x_0, \underline{e}, \overline{e} \rangle$ (*triplex*): $[\underline{x}, \overline{x}] = x_0 + [\underline{e}, \overline{e}] = [x_0 + \underline{e}, x_0 + \overline{e}]$.

Rounding:

- Containment property must be preserved for the library functions (outward rounding). $\rightarrow$ FTIA-1 still satisfied.
- Accuracy modes...

This is the easy part, on which most people agree, but...

Difficulties

Choices to standardize:

- On the mathematical model side:
  - **sqrt**(**x**) on $\mathbf{x} = [-1, 4]$? $\rightarrow [0, 2]$? Exception (error)? $[0, 2]$ with "decoration"?
  - Non-arithmetic operations: comparisons, interval-to-number operations like midpoint (pb: empty set, unbounded intervals, rounding), set operations?

Difficulties

Choices to standardize:

- On the mathematical model side:
  - **sqrt**(**x**) on $\mathbf{x} = [-1, 4]$? $\rightarrow [0, 2]$? Exception (error)? $[0, 2]$ with "decoration"?
  - Non-arithmetic operations: comparisons, interval-to-number operations like midpoint (pb: empty set, unbounded intervals, rounding), set operations?
- On the implementation (e.g. language) side: conversion of a floating-point number into an interval, e.g. `num2interval(0.1)`, where 0.1 is a decimal floating-point constant in the binary64 type?

Difficulties

Choices to standardize:

- On the mathematical model side:
  - **sqrt**(**x**) on $\mathbf{x} = [-1, 4]$? $\rightarrow [0, 2]$? Exception (error)? $[0, 2]$ with "decoration"?
  - Non-arithmetic operations: comparisons, interval-to-number operations like midpoint (pb: empty set, unbounded intervals, rounding), set operations?
- On the implementation (e.g. language) side: conversion of a floating-point number into an interval, e.g. `num2interval(0.1)`, where 0.1 is a decimal floating-point constant in the binary64 type?

  If c denotes the value of this constant (a binary64 number), normally 0.1 rounded to nearest, the answer would be $[c, c]$, not $[\nabla(0.1), \Delta(0.1)]$ (which would be expected by the average user).
  $\rightarrow$ **Containment property not satisfied!**

Difficulties

Choices to standardize:

- On the mathematical model side:
  - **sqrt**(**x**) on $\mathbf{x} = [-1, 4]$? $\rightarrow [0, 2]$? Exception (error)? $[0, 2]$ with "decoration"?
  - Non-arithmetic operations: comparisons, interval-to-number operations like midpoint (pb: empty set, unbounded intervals, rounding), set operations?

- On the implementation (e.g. language) side: conversion of a floating-point number into an interval, e.g. `num2interval(0.1)`, where 0.1 is a decimal floating-point constant in the binary64 type?

  If c denotes the value of this constant (a binary64 number), normally 0.1 rounded to nearest, the answer would be $[c, c]$, not $[\nabla(0.1), \Delta(0.1)]$ (which would be expected by the average user).
  $\rightarrow$ **Containment property not satisfied!**

- On the mathematical model & implementation sides: `nums2interval(2,1)`, i.e. $\{x \in \mathbb{R} \mid 2 \leq x \leq 1\}$?
  Empty? Exception (error)? Notion of Not-an-Interval ("NaI")?

The 4 Levels

Similar to the 4 levels of IEEE 754-2008:

Level 1 Mathematical model level:

- number system $\mathbb{R}$;
- set $\overline{\mathbb{IR}}$ of allowed intervals over $\mathbb{R}$ (Motion 3);
- principles of how the arithmetic operations ($+$, $-$, $\times$, $\div$, $\sqrt{}$, fma, exp, log, etc., and constants[1]) are extended to intervals.

Level 2 Interval (and number) datum level:

- a finite subset $\mathbb{T}$ of $\overline{\mathbb{IR}}$;
- a finite subset $\overline{\mathbb{F}}$ of $\overline{\mathbb{R}}$ (Motion 33);
- operations on these intervals (and constructors and numeric functions of intervals).

Level 3 Representation of a $\mathbb{T}$-interval (e.g., by two $\overline{\mathbb{F}}$-numbers, for an inf-sup type).

Level 4 Encoding (bit strings).

[1] Functions with no arguments (Level 1 Draft 4.4, §3.2.1 and §5.4.4).

Exception Handling / Decorations

How can one deal with *exceptions*, i.e., events (such as potential errors) it may be desirable to signal possibly in addition to some result?

- IEEE 754 (1985, 2008): status flags and traps.
- Vienna proposal (2008): status flags.
- But some people do not want flags or traps.

Note about status flags

In IEEE 754-1985, the scope of status flags is unspecified. Flags are not necessarily "global" (ambiguous notion without the context: thread? program? machine?).

In IEEE 754-2008 (§7.1), the scope of status flags is still unspecified, but it should be specified by the language or by the user.

Exception Handling / Decorations [2]

More than exceptions, one may also want information on the function f that is being evaluated, e.g.

- when f is *defined* (useful for graphics rendering);
- when f is *defined and continuous* (Brouwer's fixed-point theorem: if f is defined and continuous on $\mathbf{x}$ and $\mathbf{f}(\mathbf{x}) \subseteq \mathbf{x}$, then $\exists x \in \mathbf{x},\ f(x) = x$).

Discussions started with...

Exception Handling / Decorations [3]

```
Date: Wed, 13 May 2009 23:04:06 +0200
From: Vincent Lefevre <vincent@vinc17.org>
To: STDS-1788@LISTSERV.IEEE.ORG
Subject: Re: A proposal for the next motion

On 2009-05-13 13:18:27 -0400, Nate Hayes wrote:
> > - Example. With these definitions
> > xx*{0} = {0} and xx/{0} = Empty for any nonempty interval xx.
> > - All interval functions used here are automatically defined for all arguments
> > (e.g. sqrt([-1,4]) = [0,2], sqrt([-2,-1]) = Empty). It is left open what
> > exceptional action may be taken by an implementation on evaluating ff(ss),
> > when ff is an extension of a point function f, and ss is not a subset of D_f.
>
> These interpretations are not compatible with modal intervals.
>
> By default, they should be:
>     sqrt([-1,4]) = NaI
>     sqrt([-2,-1]) = NaI
> where NaI simply means "undefined" or "invalid operation," so that the NaI will
> be guaranteed to propagate through the remainder of a lengthy computation.

I disagree that one should get NaI in such cases. There may be a "possibly-invalid"
flag either associated with the returned interval or global (to a block or whatever)
to propagate this information if need be.
[...]
```

Exception Handling / Decorations [4]

Discussion between Nate Hayes, Vincent Lefèvre and Dan Zuras in the P1788 list in May 2009:

- No status flags, contrary to IEEE 754.
  - Nate: bad for multi-core, multi-threaded environments. (?)
  - Problems with functional languages (no global state). Dan: Forth used a tag.
- Nate: 2 different *opcodes* (operators), e.g. **sqrt**($[-1, 4]$) returns either $[0, 2]$ or NaI (note: *not* a dynamic mode).
- Vincent and Dan: no need for 2 opcodes. $\rightarrow$ Tag.
- Dan suggested various tags (e.g. inaccurate, definedButPossiblyDiscontinuous in the Vienna 3.4 document).
- Possible alignment problems with the format (not for x87 extended precision and multiple precision), e.g. 17 bytes, possible inefficiency, but optimizations and compiler directives (an application may need only one of the 2 fields).
- M. Nehmeier and J. Wolff v. Gudenberg, 2009: expressions with tags in C++ (filib++ class library).

Exception Handling / Decorations [5]

Private discussion between Vincent Lefèvre and Dan Zuras in June-July 2009.

- Notion of 3 possible states for a tag (true, false, unknown).
- Dan initially proposed many tags, too many, without always a clear meaning (e.g. missingData).
- Behavior of selection? For instance, with C's syntax:
  - if (condition) r = x; else r = y;
  - r = condition ? x : y;
- Warning with constant point functions $f(x) = c$. For an interval extension, one usually has $\mathbf{f}(\mathbf{x}) = \{c\}$, but one should have $\mathbf{f}(\varnothing) = \varnothing$.

September 2009: *decoration* (Motion 8.02).

- Notions of *decorated intervals* / *bare intervals* / *bare decorations*.
- A *decoration trit*: + (true), − (false), 0 (unknown).
- Candidates: valid (see *constructors*), defined, continuous, tight; and quite useless candidates: bounded, standard, empty, entire.
- Motion 8.02 (*Exception Handling*) passed on 27 November 2009.

More About *Trits* – Inconsistencies?

While I was writing these slides...

Two different notions of trits:

- *Certainty information* (true / false / unknown).
  For instance, for a function f on an interval $\mathbf{x}$:
  1. f is defined everywhere: $\forall x \in \mathbf{x}$, $f(x)$ is defined;
  2. f is not defined everywhere: $\exists x \in \mathbf{x}$, $f(x)$ is *not* defined;
  3. we don't know whether (1) is true or (2) is true (see *propagation*).

  Warning! Possible difference between the Level-1 interval and the (enclosing) Level-2 interval.

- *Boolean values over a set* (interval): always true / never true / sometimes true, sometimes false.
  For instance, for "sqrt(x) is defined":
  1. **sqrt**($[0, 1]$): isDefined;
  2. **sqrt**($[-2, -1]$): notDefined;
  3. **sqrt**($[-2, 1]$): possiblyDefined.

More About *Trits* – Inconsistencies? [2]

What Motion 8.02 and its rationale say...

```
1.2. A bare decoration is a list of decoration trits with possible
values +, -, and 0, characterizing part of the history of a computation
(see the rationale for possibly useful decoration trits). The values +
and - of a decoration trit make opposite certainty claims about an
associated property; the value 0 indicates the lack of certainty about
the property. A "new" standard interval created from a constructor has
a no-0 decoration of the appropriate form. The all-0 decoration is least
informative.

3.4. Useful candidates for decoration trits (possibly listed in order of
importance) are:
      isValid           possiblyValid           notValid
      isDefined         possiblyDefined         notDefined
[...]
The isValid trit might represent either an invalid construction or
uninitialized data. The isDefined trit indicates if an operation was
probed outside its natural domain, and isContinuous indicates if all
operations in the history were continuous on their argument(s), e.g.,
    [1,1]/[0,0] = (Empty,notDefined)
    [1,1]/[-1,1] = (Entire,possiblyDefined,notContinuous).
```

From *Trits* to *Tetrits*

April 2010: *trit* → *tetrit* (Dan Zuras, then Nathan Hayes) (Motion 18).

- A pair (P^+, P^-) of two opposite propositions, i.e.

$$P^+ \iff \exists x \in \mathbf{x},\ P(x),$$
$$P^- \iff \exists x \in \mathbf{x},\ \neg P(x).$$

- 4 possible values (F = false, T = true):

(P^+, P^-)	(F, F)	(F, T)	(T, T)	(T, F)
$\{P(x) \mid x \in \mathbf{x}\}$	$\varnothing$	$\{F\}$	$\{F, T\}$	$\{T\}$

  Second line: multi-valued logic on P.

- A tetrit is (F, F) iff $\mathbf{x}$ is empty. This is the difference between trits and tetrits.
- Motion 18 passed on 9 July 2010.

But contradictions with the propagation rules...

Decorations: Propagation / Tracking

Related notions: propagation, stickiness, tracking, composition of functions.

In practice, a simple program or expression, i.e. a sequence of operations, or more precisely, a computation tree (a DAG if variables can be reused, but semantically equivalent to a tree). New functions, based on existing ones.

- Behavior and meaning of decorations in an expression?
- What are the input and the function (for decorations like isDefined)?

Many discussions, but no motions passed:

- Motion 15 (April-May 2010): stickiness (similar to the IEEE 754 status flags). Not convincing in practice. Better solutions? Withdrawn.
- Competing motions (May-August 2011), which dropped *trits*/*tetrits*:
  - 25 then 25-A1 *Property Tracking*, by Nate Hayes. Propagation: *quality order*. Disagreements. Withdrawn: unified with Motion 27, to give 27-A1.
  - 26, by Neumaier & Pryce. Explicit decorations: ein/bnd/dac/def/con/emp/ill. *Containment partial order* and *propagation order*. Complex, difficult to check, apparently flawed. Problems with empty input. Practical interest? Failed.
  - 27 then 27-A1 *KISS-decorations*, by J. Wolff von Gudenberg. 4 decorations. Flawed / unclear. Too simple? Too complex? Withdrawn to be improved.

Decorations: Tracking vs Static

Mail from Dan Zuras on 25 May 2011:

> There seem to be two schools of thought about the meaning of decorations.
>
> There is the TRACKING school in which decorations are the maximal (most pessimistic) result of the tree of evaluations that led up to the result to which they are attached. That is, every exceptional or noteworthy incident in that tree is recorded for all to see whether it is relevant to the final result or not.
>
> Then there is the STATIC school in which decorations are information concerning the current result only. Earlier decorations may pass through to this result if they still apply & may be discarded if they do not. In this case the decoration must be able to be interpreted in the context of the final result whatever happened before.

Propagation of Decorations: Contradictions & Ambiguities

Motion 18 on *tetrits*: contradictions on the empty set. For instance:

$$([1,2],(T,F)) \cap ([3,4],(T,F)) = (\varnothing,(T,F))$$

(the reason of this unexpected result is that it doesn't make sense to consider the domain of a non-arithmetic expression, e.g. when $\cap$ and/or $\cup$ are involved).

Propagation of Decorations: Contradictions & Ambiguities

Motion 18 on *tetrits*: contradictions on the empty set. For instance:

$$([1,2],(T,F)) \cap ([3,4],(T,F)) = (\varnothing,(T,F))$$

(the reason of this unexpected result is that it doesn't make sense to consider the domain of a non-arithmetic expression, e.g. when $\cap$ and/or $\cup$ are involved).

Back to the question: What are the input and the function?

Let $\mathbf{y} = \mathbf{f}(\mathbf{x})$ and $\mathbf{z} = \mathbf{g}(\mathbf{y})$, where $\mathbf{f}$ and $\mathbf{g}$ are decorated interval extensions of point functions f and g. What should the decorations computed by $\mathbf{g}$ be?

- The decorations for g with input $\mathbf{y}$?
- The decorations for $g \circ f$ with input $\mathbf{x}$?

Goal of propagation: the decorations for $g \circ f$ with input $\mathbf{x}$, possibly both. The interpretation will depend *only* on $\mathbf{y}$ and its decorations (the other parts of the computation tree are unknown, not accessible).

$\rightarrow$ Possible problems when $\mathbf{y} \supset \text{Range}(f|\mathbf{x})$.

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Let $\mathbf{f}$ be a Level-2 compatible interval extension of f, i.e. with rounding.

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Let $\mathbf{f}$ be a Level-2 compatible interval extension of f, i.e. with rounding.

- $\mathbf{y} = \mathbf{f}(\mathbf{x}) = ([0, \triangle(\pi/2)], (T, F))$, i.e. isDefined (everywhere defined).
- $\mathbf{z} = \mathbf{g}(\mathbf{y}) = ([0, 1], (T, T))$, i.e. possiblyDefined (partially defined).

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Let $\mathbf{f}$ be a Level-2 compatible interval extension of f, i.e. with rounding.

- $\mathbf{y} = \mathbf{f}(\mathbf{x}) = ([0, \triangle(\pi/2)], (T, F))$, i.e. isDefined (everywhere defined).
- $\mathbf{z} = \mathbf{g}(\mathbf{y}) = ([0, 1], (T, T))$, i.e. possiblyDefined (partially defined).

However $\mathrm{Domain}(g \circ f) = [-1, 1]$, thus one would expect isDefined.

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Let $\mathbf{f}$ be a Level-2 compatible interval extension of f, i.e. with rounding.

- $\mathbf{y} = \mathbf{f}(\mathbf{x}) = ([0, \triangle(\pi/2)], (T, F))$, i.e. isDefined (everywhere defined).
- $\mathbf{z} = \mathbf{g}(\mathbf{y}) = ([0, 1], (T, T))$, i.e. possiblyDefined (partially defined).

However $\text{Domain}(g \circ f) = [-1, 1]$, thus one would expect isDefined.

Is this an error?

Propagation of Decorations: an Example

Example: $\arcsin(\frac{2}{\pi}\arcsin(\mathbf{x}))$ on $\mathbf{x} = [0, 1]$?

- $y = f(x) = \arcsin(x)$
- $z = g(y) = \arcsin(\frac{2}{\pi}y)$

Let $\mathbf{f}$ be a Level-2 compatible interval extension of f, i.e. with rounding.

- $\mathbf{y} = \mathbf{f}(\mathbf{x}) = ([0, \triangle(\pi/2)], (T, F))$, i.e. isDefined (everywhere defined).
- $\mathbf{z} = \mathbf{g}(\mathbf{y}) = ([0, 1], (T, T))$, i.e. possiblyDefined (partially defined).

However $\text{Domain}(g \circ f) = [-1, 1]$, thus one would expect isDefined.

Is this an error? Not really... Order on the decorations:

(P^+, P^-)	(F, F)	(F, T)	(T, T)	(T, F)
$\{P(x) \mid x \in \mathbf{x}\}$	$\varnothing$	$\{F\}$	$\{F, T\}$	$\{T\}$
Priority	0	1	2	3

Real decoration $\geq$ computed decoration. Related to *disjoint* vs *nested* approach.

More on Decorations

In September-October 2011:

- John Pryce's slides from his presentation on decorated intervals at the Dagstuhl-Seminar 11371 in September 2011:

  `http://www.dagstuhl.de/mat/index.en.phtml?11371#Pryce,%20John%20D.`

  `http://www.dagstuhl.de/mat/Files/11/11371/11371.PryceJohn.Slides.pdf`

- Position paper *Decorations as State Machine* by Nate Hayes:

  `http://grouper.ieee.org/groups/1788/email/pdf38EOGnbvjb.pdf`

  Disjoint approach.

- Position paper by John Pryce, but not public yet.

  Re-interpretation of Nate Hayes's *DSM* paper. Covers *computed* vs *true* decoration (difference due to partial knowledge), and mentions *nested* vs *disjoint* approach.

And discussions...

Comparison Relations

Motions:

- 13.04 (passed). 7 relations: equal ($=$), subset ($\subseteq$), less than or equal to ($\leq$), precedes or touches ($\preccurlyeq$), interior to, less than ($<$), precedes ($\prec$).

  Conditions on the bounds, but some errors with infinite bounds.
  Special rules for the empty set.

  No set-theoretic/topological definitions.

  Vincent Lefèvre: contradictions on the rules for the empty set between this motion and Motion 31 (proposed Level 1 text). See next slide.

Comparison Relations

Motions:

- 13.04 (passed). 7 relations: equal ($=$), subset ($\subseteq$), less than or equal to ($\leq$), precedes or touches ($\preccurlyeq$), interior to, less than ($<$), precedes ($\prec$).

  Conditions on the bounds, but some errors with infinite bounds.
  Special rules for the empty set.

  No set-theoretic/topological definitions.

  Vincent Lefèvre: contradictions on the rules for the empty set between this motion and Motion 31 (proposed Level 1 text). See next slide.

- 20 (failed). *Binary Relation Algebra* to build more comparison operators.

Comparison Relations

Motions:

- 13.04 (passed). 7 relations: equal ($=$), subset ($\subseteq$), less than or equal to ($\leq$), precedes or touches ($\preccurlyeq$), interior to, less than ($<$), precedes ($\prec$).

  Conditions on the bounds, but some errors with infinite bounds.
  Special rules for the empty set.

  No set-theoretic/topological definitions.

  Vincent Lefèvre: contradictions on the rules for the empty set between this motion and Motion 31 (proposed Level 1 text). See next slide.

- 20 (failed). *Binary Relation Algebra* to build more comparison operators.

- 21.2 (passed). Interval overlapping relations: before, meets, overlaps, starts, containedBy, finishes, equal, finishedBy, contains, startedBy, overlappedBy, metBy, after.

  Conditions on the bounds.

Comparison Relations: Remarks

Vincent Lefèvre, 27 January 2012:

- I wonder whether $\mathbf{x} \preccurlyeq \mathbf{y}$ and $\mathbf{x} \prec \mathbf{y}$ should be true if $\mathbf{x}$ and/or $\mathbf{y}$ is empty, because these relations should respectively be equivalent to:

  $$\forall x \in \mathbf{x},\ \forall y \in \mathbf{y},\ x \leq y \text{ (resp. } x < y\text{)}.$$

  But they would no longer be transitive on $\overline{\mathbb{IR}}$ (still transitive on $\overline{\mathbb{IR}} \backslash \{\varnothing\}$).

- Motion 13.04: $\varnothing \leq \varnothing$ is false. Motion 31: $\varnothing \leq \varnothing$ is true. In the latter case, $\mathbf{x} \leq \mathbf{y}$ is equivalent to:

  $$\forall x \in \mathbf{x},\ \exists y \in \mathbf{y},\ x \leq y,$$

  and to:

  $$\forall y \in \mathbf{y},\ \exists x \in \mathbf{x},\ x \leq y.$$

Comparison Relations: More Remarks

→ Private discussion with John Pryce in April 2012.

```
There are problems for containedInInterior and less with unbounded
intervals. I would say that their definitions are correct, e.g.
containedInInterior(Entire,Entire) and less(Entire,Entire) are
true, but the properties on the bounds should be modified.
But saying that -inf < -inf and +inf < +inf for all properties
should be sufficient (perhaps have a different notation for this
special <).
```

Proposed change by John Pryce:

- Correction of errors and for the empty set.
- Set-theoretic definitions.
- New terminology and new notation.

Number Formats

17 April 2012: Motion 33 (passed on June 6, the latest motion).

An argument or a result of some operations covered by the standard can be a number. A number is defined at Level 1 as being any member of the set $\overline{\mathbb{R}}$ of extended reals. This motion defines the corresponding notion at Level 2, called *number format* and denoted here $\overline{\mathbb{F}}$.

- Inf-sup interval type: the bounds are the members of a number format $\overline{\mathbb{F}}$.
- An implementation must support at least one number format $\overline{\mathbb{F}}$.

Not to be too strict in order to allow conventional formats, such as:

- floating-point numbers (with or without subnormals),
- fixed-point numbers (including integers),
- rational numbers (with bounded numerator and denominator),
- double-double numbers (such as provided by the `long double` C type in the current PowerPC ABI).

Number Formats [2]

Requirements and recommendations (with rationale):

A1 $\overline{\mathbb{F}}$: finite subset of $\mathbb{R}$ with 3 special datums: $-\infty$, $+\infty$ (both in $\overline{\mathbb{R}}$) and NaN.
Non-ambiguous rounding; smallest and largest finite elements (for midpoint).

A2 Different variants/representations allowed, not distinguished in $\overline{\mathbb{F}}$, possibly except for the number 0, which may be regarded as signed: $+0$ and -0.
Signed zero: for 754-conforming implementations. Like in ISO C and LIA-2.

A3 $0 \in \overline{\mathbb{F}}$ (or $+0, -0 \in \overline{\mathbb{F}}$).
Probably needed for some functions (e.g. smallest radius).

A4 The format must be *symmetric*, i.e. if a real x is in $\overline{\mathbb{F}}$, then $-x$ is also in $\overline{\mathbb{F}}$.
Maybe not really necessary, but expected by the user. See known formats.

A5 Definition of *rounding*. Possible constraint on the rounding direction.
Recommended: *correct rounding*, else documented *error bound*.
Correct rounding desirable for accuracy, but only recommended for efficiency reasons, in particular with exotic number formats. But see B4.

Number Formats [3]

A *754 format* of a 754-conforming implementation: one of the number formats defined in IEEE 754 with the following binding to A1-A5.

- B1 The infinities correspond to the IEEE 754 infinities.
- B2 NaN corresponds to the IEEE 754 Level 2 NaN (Level 3: qNaN for results).
- B3 The number 0 is signed.
- B4 At least one number format (for inf-sup, the one of the interval bounds) must follow the IEEE 754 rules (correct rounding).

Notion of number format $\overline{\mathbb{F}}$ *compatible* with an interval type $\mathbb{T}$: require $\overline{\mathbb{F}}$ to be dense enough in order to satisfy some properties when returning a numeric value.

- C1 For each non-empty interval $\mathbf{x}$ of type $\mathbb{T}$ (Level 2), there exists a finite number x of $\overline{\mathbb{F}}$ such that $x \in \mathbf{x}$.
  One can define a Level 2 midpoint that belongs to any non-empty $\mathbb{T}$-interval.

For some specific operations involving $\mathbb{T}$-intervals and $\overline{\mathbb{F}}$-numbers (to be decided by later motions), $\overline{\mathbb{F}}$ shall be compatible with $\mathbb{T}$ (but shouldn't if this isn't really necessary).

Level 1 Constructors

→ To create an interval from numbers or a text string (Motion 30.02).

- `nums2bareinterval(l,u)` / `nums2interval(l,u)`, which returns $[l, u] = \{x \in \mathbb{R} \mid l \leq x \leq u\}$ if $l \leq u$, $l \neq +\infty$ and $u \neq -\infty$, otherwise undefined.
- `text2bareinterval(t)` / `text2interval(t)`, which returns an interval according to some predefined syntax, otherwise undefined.
- `bareempty()` / `empty()`, which returns Empty ($\varnothing$).
- `bareentire()` / `entire()`, which returns Entire ($\mathbb{R}$).

No longer `num2bareinterval(x)` / `num2interval(x)`, which were equivalent to `nums2bareinterval(x,x)` / `nums2interval(x,x)`, but dangerous if `x` was computed in floating-point.

No Level 2 yet, but it will be based on the number formats.

Numeric Functions of Intervals

In the current Level 1 draft of the P1788 standard text, before any motion.

Disagreements on particular cases (empty set, unbounded intervals), in particular at Level 2.

- `inf`: lower bound $\underline{x}$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? $+\infty$?
- `sup`: upper bound $\overline{x}$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? $-\infty$?
- `mid`: midpoint $(\underline{x} + \overline{x})/2$ of $\mathbf{x}$. Undefined at Level 1 if $\mathbf{x} = \varnothing$ or is unbounded.
- `wid`: width $\overline{x} - \underline{x}$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? 0? $-\infty$?
- `rad`: radius $(\overline{x} - \underline{x})/2$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? 0? $-\infty$?
- `mag`: magnitude $\sup\{|x| \mid x \in \mathbf{x}\}$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? 0? $-\infty$?
- `mig`: mignitude $\inf\{|x| \mid x \in \mathbf{x}\}$ of $\mathbf{x}$. For $\mathbf{x} = \varnothing$: undefined? $+\infty$?

Discussions followed, in particular on the Level 2 midpoint...

Midpoint

Midpoint and unbounded intervals: The midpoint operator can be specified so that it can be used for interval splitting (in branch-and-bound algorithms). But for large (bounded) intervals, the midpoint is a poor choice.

Arnold Neumaier: two functions `mid` and `flmedian`?

Dan Zuras: midpoint for non-Empty intervals at Level 1 and Level 2 (Motion 32).

At Level 2, in addition to unbounded intervals, take overflow into account:

- $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathbb{R}) = 0$ due to Level 2 symmetry.
- If $\circ_{\overline{\mathbb{F}}}((\underline{x} + \overline{x})/2) = +\infty$, then $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathbf{x}) = \mathrm{nextDown}_{\overline{\mathbb{F}}}(+\infty)$.
- If $\circ_{\overline{\mathbb{F}}}((\underline{x} + \overline{x})/2) = -\infty$, then $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathbf{x}) = \mathrm{nextUp}_{\overline{\mathbb{F}}}(-\infty)$.

Note: as a consequence, with correct rounding for each operation, $\inf_{\overline{\mathbb{F}}}(\mathbf{x}) \leq \mathrm{mid}_{\overline{\mathbb{F}}}(\mathbf{x}) \leq \sup_{\overline{\mathbb{F}}}(\mathbf{x})$.

Motion 32 failed (12 *yes*, 30 *no*). Partially flawed...

Midpoint [2]

```
From: Vincent Lefevre <vincent@vinc17.org>
Subject: Re: I vote NO on: Motion P1788/M0032:midpoint
Date: Mon, 2 Apr 2012 15:04:28 +0200
```

The motion says that for any X and Y that live at Level 2,

```
X \subset Y ==> if (inf_F(X)==inf_F(Y)) then mid_F(X) <= mid_F(Y)
```

Let $\overline{\mathbb{F}}$ be a radix-10, 1-digit precision floating-point system. Let the interval type be based on a triplex representation $\langle \mathrm{mid}, \mathrm{rad1}, \mathrm{rad2} \rangle$. Let's choose:

- $\mathtt{X} = \langle 20, -5, -3 \rangle = [15, 17]$
- $\mathtt{Y} = \langle 20, -9, -3 \rangle = [11, 17]$

$\mathtt{X} \subseteq \mathtt{Y}$ and $\inf_{\overline{\mathbb{F}}}(\mathtt{X}) = \inf_{\overline{\mathbb{F}}}(\mathtt{Y}) = 10$, so that the conditions are satisfied. Then:

- $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{X}) = \circ_{\overline{\mathbb{F}}} \mathrm{mid}(\mathtt{X}) = \circ_{\overline{\mathbb{F}}}(16) = 20$,
- $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{Y}) = \circ_{\overline{\mathbb{F}}} \mathrm{mid}(\mathtt{Y}) = \circ_{\overline{\mathbb{F}}}(14) = 10$.

The inequality $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{X}) \leq \mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{Y})$ doesn't hold! Actually, I don't see anything wrong by having $\mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{X}) > \mathrm{mid}_{\overline{\mathbb{F}}}(\mathtt{Y})$ here, since $\mathtt{X} \geq \mathtt{Y}$.
The problem comes from the fact that the midpoint function is based on a Level 1 definition while the property is based on Level 2 lower bounds of the intervals.