Constructors

Jürgen Wolff von Gudenberg, Marco Nehmeier
University of Würzburg

April 19, 2011

Rationale

When defining constructors for decorated intervals the mathematical rigor which clearly dominates the work of P1788 should be composed with some common sense rules. These rules shall support the acceptance of the standard. Hence in this position paper, we reiterate a discussion from the mailing list and propose a solution that follows the "Law of Least Surprise" and thus is acceptable for every user. Nevertheless it delivers the rigorous results in all cases where the input is finite. Proper supersets are computed for infinite constructor inputs.

At first we consider suggested constructors prior to decorations. They were modeled as independent functions whereas we will use overloaded function names and provide one main constructor that is able to build (all) the other ones. Please note that x and y denote floating point values as well as s represents a text string.

1 Existing Proposals

1.1 Vienna Proposal

There are constructors for standard, as well as for non-standard intervals.

1. standardInterval(x, y)
2. anyInterval(x, y)
3. Entire()
4. Empty()

Furthermore there are two functions for the construction of mid-rad intervals intervalAbs(x,d) and intervalRel(x,d)

Type conversions between text strings and intervals are treated by explicit syntax rules for the conversion as well as the function number2Interval(x). Implicit type conversion is enabled in mixed type expressions.

1.2 P1788 Draft Standard

The current draft 02.2 provides constructors in section 5.4.1. That draft version will soon be replaced by version 03.1

5 constructors are specified.

1. floatsToInterval(x, y)
2. floatToInterval(x)
3. textToInterval(s)
4. Entire()
5. Empty()

Here s denotes a text string that defines a mathematical interval.

1.3 Common pitfalls

Different language or cultural conventions should not play a role in P1788. Therefore the text to interval conversion shall not contain delimiter- or separator characters. Note that even the floating point habitually is a comma in Germany. We only need a transformation of a text string representing a real number to the datum of the underlying type that is the closest neighbor in the rounding direction (towards + or - infinity). That constructor should also accept strings for special datums, "NaN" or "infinity". For those text-to-interval constructors some of the rules are not valid, namely `Interval`("0.1") is not a point interval.

The second and main point is introduced by an e-mail by Siegfried Rump that reminds on long discussions in the past. The example given may be found in the Vienna proposal as well, but with other solutions.

```
Assume a user has already an interval exponential function and needs
an interval hyperbolic cosine, _only_ for positive arguments. With
(*)   cosh(x) = ( exp(x) + exp(-x) ) / 2
this is easy to do. The user knows that just inserting an interval X into (*)
causes overestimation. But cosh is strictly convex for positive arguments,
so
     cosh([Xinf,Xsup]) = [ cosh(Xinf) , cosh(Xsup) ]
with proper rounding is correct for positive input. Using the interval exponential
for the point intervals [Xinf,Xinf] and [Xsup,Xsup] does the job.

Let's use the function names Lbound and Ubound to access the lower and
upper bound of an interval, so Lbound(X) is Xinf and Ubound(X) is Xsup.
Furthermore call the typecast from a floating-point number to a point
```

```
interval Interval(.), so that Interval(x) yields the point interval containing x.
Also denote the union of intervals X,Y by union(X,Y).

Now the user writes the following program:

function Y = cosh(X)
    Xinf = Interval(Lbound(X))
    Xsup = Interval(Ubound(X))
    C1 = ( exp(Xinf) + exp(-Xinf) ) / 2
    C2 = ( exp(Xsup) + exp(-Xsup) ) / 2
    Y = union(C1,C2)

I claim this program looks innocent and correct.

Of course, it is not, that is the point. If the right bound Xsup of the
input interval X is infinite, then Interval(Ubound(X)) yields the empty
set. The exponential of the empty set is empty, so the final result
is just C1.

I also claim that this may easily slip well written test routines, and
the false cosh routine may be used for a long time until eventually
the error is discovered.

The most serious concern I have is that even if WE document this
behaviour very well, in a short while WE forget about this and WE
ourselves will fall into this trap. Believe me, I have this experience.

I think the only way to cure this is to redefine Interval(infinity) to
be [realmax,infinity], i.e. the set of real numbers not less than realmax.

I think the standard should manifest what everybody assumes anyway,
the default should be the most likely.
The best standard is a document which we never have to consult.
```

We try to find a compromise solution.

2 Constructors for Decorated Intervals

2.1 Decorated Intervals

Intervals are sets of real numbers (motion 3) decorated with 1 of the 5 flavors defined in the exception handling part. Formally a decoration is a pair of a

function f and an interval $\mathbf{x}$ for which the listed property is valid.

- saf: everywhere defined, bounded and continuous.
- def: everywhere defined
- con: always true
- emp: nowhere defined
- ill: empty, ill-formed

The exceptions are ordered according to quality from top to bottom.

Constructors are functions building a (decorated) interval. Not all combinations of interval and decoration part make sense.

2.2 Constructors

2.2.1 Level 1

A constructor is a (surjective) mapping $\mathbb{R}^* \times \mathbb{R}^* \times \mathbb{D} \to \overline{\mathbb{DIR}}$. Note that the set of all decorated intervals is a proper subset of the Cartesian product, since many combinations are not allowed. Even the restriction to bare intervals is not one to one (injective), because the empty set has many origins. But the restriction to non-empty intervals can be inverted. The inverse may be used to specify the powerset operation definitions as executable formulas with the constructor parameters.

2.2.2 Level 2

An abstract data type is defined in level 2. This motion is about the inf-sup representation of that type. Hence, the specification of the operations uses the endpoints. Let

$$\begin{aligned} c : \mathbb{F} \times \mathbb{F} \times \mathbb{D} &\to \overline{\mathbb{DIF}} \\ (x, y, d) &\mapsto [x, y]_d \end{aligned}$$

be the constructor function. It is not defined for $(-\infty, -\infty, d)$, (∞, ∞, d) and all triplets (x, y, d) with $x > y$. c is bounded and continuous, if both x and y are bounded. Constructor parameters are floating point datums including NaN. The inverse constructor function resolves into two functions inf, sup from $\overline{\mathbb{DIF}}$ to $\mathbb{F}$ and a function $deco : \overline{\mathbb{DIF}} \to \mathbb{D}$.

2.2.3 Default Constructor

We define the constructors with explicit access to the endpoints. Hence, the inf-sup instance of level 2 is concerned.

For each supported format P1788 shall provide a constructor with 3 arguments (x, y, d) specifying the lower or upper endpoint and the decoration. x and y are datums of the supported format, d is a decoration value. The resulting decorated interval depends on a precondition that holds between the arguments.

A constructor often is called inside a function to define the return value. In this case the decoration parameter has to reflect the possible exception.

Example:

$\sqrt{[0,1]} = Interval(0, 1, saf) = [0, 1]_{saf}$

$\sqrt{[-1,1]} = Interval(0, 1, def) = [0, 1]_{def}$

In the following table we show how the bare interval part depends on the precondition, and that several intervals have an intrinsic decoration flavor id. The output decoration is the minimum of the intrinsic and the given value, denoted by deco $= d \wedge id$. $R =$realmax denotes the overflow threshold.

argument	precondition	box	dec $d\wedge\ \ldots$	remark
(x, y, d)	$-\infty = x = y$	$[-\infty, -R]$	con	not bounded, not defined
	$-\infty = x < y < \infty$	$[-\infty, y]$	def	not bounded, but defined
	$-\infty = x < y = \infty$	$[-\infty, \infty]$	def	not bounded, but defined
	$-\infty < x \leq y < \infty$	$[x, y]$	saf	
	$-\infty < x < y = \infty$	$[x, \infty]$	def	
	$-\infty < x = y = \infty$	$[R, \infty]$	con	
	$x > y$	$[x, y]$	ill	keep the twisted bounds as a hook for extensions
	$x = NaN$ or $y = NaN$	$[NaN, NaN]$	ill	illegal emptyset

2.2.4 Interval(∞)

We have defined another explicit notation for the 2 intervals that represent the positive or negative overflow range.

Interval(∞) $= \{x \in \mathbb{R} | x \geq R\}$, Interval($-\infty$) $= \{x \in \mathbb{R} | x \leq -R\}$

Those terms otherwise would denote the empty set, but we have enough notations for that.

In any case where there is a difference, our version of the constructor contains the results found by the rigid interpretation.

Interval(1/0.0) =Interval(∞) =[R,∞] $\supset \emptyset_{emp}$= [1,1] / [0,0]

2.2.5 Further constructors

The default value for the decoration parameter is saf. 5 further constructors for the 2 parameter version are defined for convenience. Their semantics are given with respect to the main constructor. They are listed in the following table. A similar table may be provided for the 3 parameter version.

argument	semantics	remark
(x)	Interval(x,x)	point interval
("p","q")	Interval(fpdown("p"), fpup("q"))	see below
("p")	Interval(fpdown("p"), fpup("p"))	not necessarily point interval
(A)	A = A.box$_{A.dec}$	copy
()	$\emptyset_{emp}$	valid emptyset

fpdown and fpup are functions that deliver a floating point datum from a given string. They check whether the input string represents a real number or infinity. If this is not the case, NaN is returned, else they convert such a string to the adjacent floating point datum towards minus or plus infinity, respectively.

Note further that the only way to construct a valid emptyset is to use the empty argument list.

The Motion

For each supported format P1788 shall provide a constructor with 3 arguments (x, y, d) specifying the lower or upper endpoint and the decoration. x and y are datums of the supported format, d is a decoration value. The resulting decorated interval depends on a precondition that holds between the arguments, see 2.2.3

It shall also provide two functions to transform strings into floating point datums

and a constructor for a valid emptyset with an empty parameter list.

More constructors like those given in 2.2.5 should be provided.