Exception Handling for Interval Arithmetic

This motion specifies generalities about exception handling for interval arithmetic.

1. Definitions

1.1. A bare interval is either a standard interval, i.e., an element of overline-IR as specified in Motions 5 and 6, or a nonstandard interval, whose detailed specification will be a matter of a later motion. (One possible later decision might be to have no nonstandard intervals; then the current distinction between bare intervals and standard intervals would not be needed.)

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.

1.3. A decorated interval consists of exactly one interval and one decoration; it is standard iff the interval part is standard.

1.4. For simplicity, we refer to a bare interval or bare decoration just as an interval or decoration, respectively, except when the bareness is to be emphasized.

2. Motion Text

2.1. P1788 provides

- bare intervals;
- bare decorations;
- decorated intervals;
- arithmetic operations defined on intervals, decorations, and decorated intervals; and
- forgetful operations that drop either the decoration or the interval from a decorated interval.

2.2. An operation on standard decorated intervals returns a standard decorated interval whose interval is the result of the operation on the argument intervals, and whose decorations are computed from the arguments such that they retain the most informative and valid information about the interval. All result decorations will be completely specified (later) according to the intended semantics of the decoration trits.

2.3. An operation on bare decorations is obtained by promoting the bare decorations to decorated intervals whose intervals are the empty set and then performing the operation with the resulting decorated intervals.

2.4. An operation on bare intervals is obtained by promoting the bare intervals to decorated intervals whose decorations are all-0 and then performing the operation with the resulting decorated intervals (if any further promotion rules are required, they will be specified and voted on in another motion).

2.5. Forgetful operations behave as specified above except they throw away either the interval or decoration portion of the result.

3. Rationale

3.1. This motion uniformly handles all arithmetic and nonarithmetic exceptions that are relevant for interval arithmetic and its applications. It eliminates the need for separate global sticky flags, and integrates non-intervals (NaI), without introducing any overhead for users who don't make use of exceptions.

3.2. Recent discussion in the P1788 forum showed that some interval algorithms require decorated intervals while others only require intervals and/or decorations (or NaI's). There are various implementation and performance tradeoffs to be gained or lost by restricting interval computations to only one of these types of objects. These tradeoffs may depend on available (present or future) platforms, hence the choices should be left to implementors for exploitation, rather than be fixed by the standard.

The framework presented in this motion also unifies the concept of NaI and Empty in a semantically correct way. For example, given any non-empty interval X,
X \union Empty = X
is usually the case. However, depending on the history that created Empty, some applications may need
X \union Empty = Empty,
which is the same semantics as NaI, i.e.,
X \union NaI = NaI.
This can be neatly handled by decorations.

3.3. Only minimal requirements for a consistent behavior of decorations are fixed by this motion. The 3-valuedness of decoration trits is needed to have a clear way of organizing the deterioration of antagonistic information. Just as one cannot avoid overestimation in intervals, one cannot avoid getting less and less informative decorations if different decorations are to be combined. Since a bare interval has lost its decoration, it must be assumed to possibly have the worst decoration, and this will propagate when combined with a bare decoration.

3.4. Useful candidates for decoration trits are:

isValid	possiblyValid	notValid
isStandard	possiblyStandard	notStandard
isEmpty	possiblyEmpty	notEmpty
isEntire	possiblyEntire	notEntire
isBounded	possiblyBounded	notBounded
isDefined	possiblyDefined	notDefined
isContinuous	possiblyContinuous	notContinuous
isTight	possiblyTight	notTight

These fit exactly into 2 bytes if each trit is represented by two bits.

3.5. Some things this motion specifically does not do (but some of which need to be decided later):

-- define the choice, semantics, or concrete representation of the decoration trits;
-- define all details of how operations on intervals, decorations, and decorated intervals behave;
-- define the forgetful operations or how they behave;
-- define requirements for (or the presence of) nonstandard intervals;
-- define how decorated intervals are to be represented in a concrete format; and
-- define an interchange format for decorated intervals.

Regarding the last two items, a concrete representation format is specific to each implementation; the interchange format is what is written into a file for exchange with a possibly different implementation. The latter should be standardized; the former should not be to give maximal freedom to implementors. In either case, the present motion is agnostic on these issues, and such decisions will be subject of future motions.