SC22/WG11 N401

Resolution of International Comments on LIA-1

May 1994

There were 18 votes to approve LIA-1 and 2 votes to disapprove (France and Sweden). The French no vote can be reversed by correcting the French translation of the document's title. It appears that the Swedish no vote cannot be resolved without substantial normative changes. These would set the project back to the DIS or even CD stage, and would risk the loss of one or more of the current yes votes.

Comments from France

Change French title [ACCEPT]:

WG11 agrees that the current French title of LIA-1 is incorrect, and urges the ITTF to adopt the title submitted by AFNOR.

Page 58 [ACCEPT]:

We will delete this comment on Fortran.

Page 83 [ACCEPT IN PART]:

This remains a correct comment about the (now obsolete) Fortran 77 standard. We will place it in the past tense.

Page 91 [ACCEPT IN PART]:

The parentheses will be removed from the print statement. The expression involving "HUGE" seems no less portable (in a formal sense) than any other expression containing a numeric literal.

Page 92 (G3) [ACCEPT]:

The assignment symbol will be corrected.

Page 92 (G4) [ACCEPT]:

The call keyword will be added.

Annex H [ACCEPT]:

The ANSI reference will be corrected.

Additional Comments from France

Three of the five additional comments are repetitions of comments addressed above. For the other two:

Page 82 [REJECT]:

Fortran's EXPONENT function is not quite the same as LIA-1's EXPON.

Page 93 [ACCEPT]:

X will be replaced by Y as appropriate.

Comments from Sweden

The recommended changes are all substantial normative changes. Adopting them would require reballoting LIA-1 at the DIS or even CD level. Worse, adopting these recommendations would probably change at least one national vote from APPROVE to DISAPPROVE, resulting in no net improvement.

1) Add natural numbers [REJECT]:

Unbounded unsigned integers are not currently included in LIA-1 because no standard language provides such a type, and because it would require yet another parameter. However, WG11 would like to invite Sweden to propose an addendum to LIA-1 on this topic.

2) Remove modulo integers [REJECT]:

Modulo integers do have legitimate (if limited) uses. They are an important datatype in C. We have received many expert comments stating that including a definition of modulo integers is essential. (Note that languages and implementations are not required by LIA-1 to provide modulo integers.)

Additional text will be added in the rationale which describes the hazards of modulo integers, and recommends that whenever systems provide modulo integers, that they provide a corresponding non-modulo integer type as well.

3) Replace add* by + [REJECT]:

Add* is needed to describe at least one prominent floating point architecture. If the industry evolves as expected, it may be appropriate to remove add* at the first 5-year review.

4) Div/Rem/Mod [ACCEPT IN PART]:

The truncation variants of div and rem are needed for compatibility with Fortran. Removing them will not change Fortran (or any other language), but will make it just that much harder to document a language's choices.

A similar argument applies for the Pascal variant of mod.

However, additional text will be added to clause A.5.1.3 to explain the disadvantages of the truncating div and rem, and to recommend the mathematically superior versions.

The superscripts on div, rem, and mod will be changed as follows:

div^t and rem^t will be used for truncating div/rem

```
        div^f and rem^f will be used for flooring div/rem
        mod^a will be used for mod of any modulus
        mod^p will be used for mod restricted to positive modulus

    A note will be added mentioning that rem^f is equivalent to mod^a.

    Annex E will be reviewed for any inconsistencies with language
    standards.

5) Convert F to I [REJECT]:

    Several standard languages choose to leave this conversion weakly
    defined (just as LIA-1 does).  The three specifically recommended
    conversions (ceiling, floor, and nearest) are already in the first draft
    of LIA-2.


Comments from the United Kingdom
--------------------------------

Clause 4.1 [ACCEPT]:

   The word "non-empty" will be added.


Comments from the United States
-------------------------------

A1 [REJECT]:

    No rationale for this change was given, and no alternative text was
    submitted.

A2 [REJECT]:

    No rationale for this change was given, and no alternative text was
    submitted.

A3 [ACCEPT IN PART]:

    The final sentence will be rewritten as "... particular set of parameter
    values is selected, and all required documentation is supplied, the
    resulting information should be precise enough to permit careful
    numerical analysis."

A4 [ACCEPT]:

   "Characterize" will be replaced by "describe".

A5 [REJECT]:

    No explanation of the problem was given, and no alternative text was
    submitted.

A6 [REJECT]:

    The benefits section was reexamined.  No false promises were found.

A7 [ACCEPT]:
```

The phrase beginning "to determine" will be replaced by "to the programmer".

A8 [ACCEPT IN PART]:

The following clarifying sentence will be added: "However, specifications for such values are given in IEC 559."

A9 [REJECT]:

No rationale for this change was given, and no alternative text was submitted.

A10 [ACCEPT IN PART]:

A note will be added to the definition of exceptional value stating that

"Exceptional values are not to be confused with the NaNs and infinities defined in IEC 559. Contrast this definition with that of continuation value above."

A11 [ACCEPT]:

The following text will be added to the second definition of error: "Error and exception are not synonyms in any other context."

A12 [ACCEPT]:

The definition will be changed to

"Exception: The inability of an operation to return a suitable numeric result. This might arise because no such result exists mathematically, or because the mathematical result cannot be represented with sufficient accuracy."

A13 [ACCEPT]:

See resolution of A10 and A11.

A14 [REJECT]:

The existing text seems fine.

A15 [REJECT]:

No explanation of the problem was given, and no alternative text was submitted.

However, the following text will be added: "Note that a suitable representable result may not exist (see 5.2.6)."

A16 [WITHDRAWN]:

The U.S. indicated that their real concern was the completeness of information available at runtime. This is better expressed in A26.

A17 [REJECT]:

No additional changes seem to be needed.

A18 [ACCEPT IN PART]:

The text will be changed to "Whenever an arithmetic operation (as defined in this clause) returns ...". The operations in clause 5 return exceptional values in order to signify that a notification is required.

A19 [WITHDRAWN]:

The U.S. has reconsidered this recommendation.

A20 [ACCEPT]:

The phrase "unbounded extension" will be used instead.

A21 [ACCEPT]:

A new definition of signature will remove this problem. The definition of signature will be changed to:

"Signature (of an operation or function): A summary of information about an operation or function. A signature includes the operation name, the minimum set of inputs to the operation, and the maximum set of outputs from the operation (including exceptional values if any). The signature

```
    add_I:  I x I --> I u {integer_overflow}
```

states that the operation named add_I shall accept any pair of I values as input, and (when given such input) shall return either a single I value as its output or the exceptional value integer_overflow.

A signature for an operation or function does not forbid the operation from accepting a wider range of inputs, nor does it guarantee that every value in the output range will actually be returned for some input. An operation given inputs outside the stipulated input range may produce results outside the stipulated output range."

In addition the following note will be added to 5.2.2:

"NOTE -- Operations are permitted to accept inputs not listed above. In particular, IEC 559 requires floating point operations to accept infinities and NaNs as inputs. Such values are not in F."

A22 [REJECT]:

This issue has been addressed before. The problem is that no one can come up with suitable text. No alternative text was offered by the U.S.

A23 [ACCEPT IN PART]:

The clauses will be rewritten to replace "shall" forms with "is" forms. The word "requirements" will be avoided. The grammar of the final sentence in 5.2.5 will be improved.

A24 [ACCEPT]:

The reference will be corrected.

A25 [ACCEPT]:

We will alter the sentence to read "... is not specified by this definition."

A26 [REJECT]:

Any IEC 559 binding will include a way to find out which of the four IEC 559 rounding modes is currently selected. As for rnd_style itself, the required values should not include non-conforming modes (the directed roundings), and the two forms of nearest can be distinguished (in practice) by looking at iec_559.

A27 [REJECT]:

The U.S. did not explain why they feel this clause is incomplete, and no alternative text was submitted.

A28 [ACCEPT]:

The sentence will be changed to "different choices for rnd_F->I or nearest_F can produce different conversion operations."

A29 [ACCEPT IN PART]:

After extensive discussion, the problem seems to be that readers are not remembering the provisions of clause 2 while they are reading 6.1.1. Thus, they do not realise that an exception handling mechanism defined in a language or binding standard will take precedence over the mechanisms introduced in clause 6.

To avoid this possible misreading, the title and first three sentences of 6.1.1 will be replaced as follows:

> 6.1.1 Language defined notification
>
> If the programming language in use defines an exception handling mechanism that can
> a) detect the occurrence of arithmetic exceptions,
> b) report such exceptions to the executing program,
> c) permit the programmer to specify to compensate for such exceptions, and then
> d) continue program execution,
>
> then notifications shall be handled by that language defined mechanism.
>
> Such a mechanism may be defined as part of the programming language standard itself or by a separate binding standard.
>
> > NOTE -- The exception handling mechanisms of Ada and PL/I are examples of language defined notification. In these languages, an exception causes a prompt alteration of control flow to execute user provided exception handling code. Other notification mechanisms, such as continued execution with special non-numeric "error values", may be appropriate for other languages.

This rephrases the current requirements in a clearer and more explicit form, but does not add anything new.

Add to the first note on p.22: "If the iec_559 parameter is true, the continuation values must be precisely those stipulated in IEC 559."

A30 [REJECT]:

The U.S. did not explain why they feel this clause is too vague, and no alternative text was submitted.

B1 [ACCEPT]:

Accepted as written.

B2 [ACCEPT]:

Accepted as written.

B3 [ACCEPT]:

Accepted as written.

B4 [ACCEPT]:

Accepted as written.

B5 [ACCEPT IN PART]:

We will add (unsigned int) and (unsigned long) to the list of C conversions on page 75. However, it became clear during discussions that the subsequent paragraph is correct as it stands.

B6 [ACCEPT]:

Accepted as written. Use boldface for "constant", "while", "abs", "loop", and "end" if practical.

B7 [REJECT]:

This is really directed at language designers. However, WG11 feels that language designers already understand this issue.

B8 [ACCEPT]:

Add the following to annex C para 2: " A programming language binding for a standard such as IEC 559 must define syntax for all required facilities, and should define syntax for all optional facilities as well. Defining syntax for optional facilities does not make those facilities required. All it does is ensure that those implementations that choose to provide an optional facility will do so using a standardized syntax."

In the next sentence, change "those facilities" to "all IEC 559 facilities." Throughout the annex, weaken "must" to "should."

B9 [ACCEPT IN PART]:

These experts (and others) should be consulted, but preference should be given to the development of normative binding standards, not polishing the examples in LIA-1.

C1 [ACCEPT]:

WG11 agrees with the U.S. on this point.

Comments from Individual Experts

A number of comments were received from individual experts. Those comments that led to changes in the text are listed here.

Page vii, last sentence (Scowen): Modify the sentence to read "the results are reliable if and only if there is no notification."

Page 7 (Schaffert): Add text to explicitly allow a language to include non-numeric values in their F, and to allow additional operations.

Page 7 (Karlsson): Use italic I when introducing maxint and minint.

Page 15 (Kulisch and Walther): Delete ";" after F*.

Page 17 (Kulisch and Walther): In the first note, replace "lower bounds" with "smaller values for rnd_error."

Page 21 (Karlsson): Change "save_indicators" to "current_indicators."

Page 23 (Kulisch and Walther): Add Fortran to the list of languages that use == for equal.

Page 24 (Gay): Change "transformation" to plural.

Page 28 (Kulisch and Walther): Clarify that K-M arithmetic is entirely compatible with LIA-1's requirements, but is stricter.

Page 28 (Kulisch and Walther): Cite the following in addition to [33]:
U. Kulisch and W. L. Miranker: Computer Arithmetic in Theory and Practice, Academic Press, New York, 1981

Page 39, Annex A.5.2.0.4 (Eggert): Add to the second para: "However, be sure to read 5.2.9 and C.1."

Page 44 (Karlsson): Italicize two occurrences of x.

Page 55, first para (WG11): Change "portable" to "standard"

Page 59 (Karlsson): Delete rnd_I from middle of page.

Page 61, 4th para (Schaffert): Change "requiring full accuracy" to "requiring full conformity to LIA-1".

Page 64, Annex C.2 (Eggert): Change the spelling of divide-by-zero to divide_by_zero.

Page 68 (Karlsson): Cut "where x is an expression of type FLT"

Page 69 (Karlsson): rem^1 is misspelled as mod.

Page 69 et seq (Karlsson): When a binding for mod^1 exists, use it for rem^1 as well. (Bindings effected: Ada, Fortran, PL/I)

Page 73, Annex E.4 (Eggert): Prepend "Integer valued" to the sentence "Parameters and derived constants can be used in preprocessor expressions."

Page 74 (Jones): Alter the FLT_ROUNDS table as follows:

```
    truncate    FLT_ROUNDS = 0
    nearest     FLT_ROUNDS = 1
    other       FLT_ROUNDS /= 0 or 1
```

Remove "and subtraction" from the subsequent note.

Page 75 (Karlsson): Replace "floor" with "intprt".

Page 76 (Jones): Use "|" for combining indicators.

Page 78 (Karlsson): Swap Lisp "rem" and "mod" as bindings for the two kinds of rem_I.

Page 84 (WG11): Restore the example binding for Pascal, appropriately updated. Check other bindings against current standards and update as needed.

Page 84 (Karlsson): Add "integer k".

Pages 84 to 87 (Klensin): Change PL/I references to
ANSI X3.74, PL/I General Purpose Subset
ISO 6522-1993, General Purpose PL/I

The editor is authorized to consult with Klensin and make appropriate modifications to the PL/I binding example to conform to the above standards.

Page 88 (WG11): Also reference IEEE 754.

Page 89, Annex F.4 (Eggert): Replace "defined in 754" with "implied by 754"

Page 90, Annex F.5 (WG11): Add a reference to the proper section in the Fortran standard. Remove the citation [1]. Add an example discussion of extended precision intermediate values.

Page 91 to 94 (Karlsson): Align closing clauses as is common in Fortran.

Page 93, G.5 and G.6 (Eggert): Explain that exception are assumed not to arise, or are being omitted for clarity.

Page 95, annex H (Kulisch): Update reference [3] to the official ISO and ANSI standards.

Page 95, ref [9] (Jones): Cut "First Edition" here and in refs. Lengthen hyphens.

Page 97, annex H (Kulisch): Add " (eds.)" after "Miranker" in reference [33].