Final Minutes of 7th ARG Meeting
7-9 October 1998
Paris, France

Attendance: John Barnes, Randy Brukardt, Robert Dewar (third day), Gary Dismukes, Pascal Leroy, Erhard Ploedereder, Jean-Pierre Rosen, Tucker Taft

Meeting Summary

The meeting convened on 7 October at 9:00 a.m. at the Rational Offices in Paris/Montigny, and adjourned at 2:15 p.m on 9 October.

Roughly one day and a half were spent on AIs, a half-day on future extensions, and the remainder on the draft of the conformance standard, from section 5 to the end of the document. The details on these topics are found in later sections of these minutes, which, as usual, are not in chronological order.

The order of discussion items on the agenda was significantly influenced by a strike of the local transportation system, which caused unpredictable delays in arrivals and early departures.

The ARG unanimously thanked Pascal and Rational profusely for the generous hosting of the meeting.

AI Summary

The AIs were discussed during the meeting in nearly the same order as listed in the meeting agenda and (more than) the whole agenda was covered.

The following AI were **approved** without changes:

AI-119 - Is Normal Termination an "External Interaction"? (9-2-0)
AI-206 - Ada.Task_Identification.Is_Callable for the environment task (7-0-0)

The following AIs were **approved with small changes**, soon ready for editorial review:

AI-164 - Definition of remote access type (7-0-0)
AI-166 - Parameterless_Handler values designate default treatment (7-0-0)
AI-168 - Aliased objects can not have discriminants modified (7-0-0)
AI-184 - View conversion to an indefinite subtype (7-0-0)
AI-190 - Compile-time vs. Run-time Errors (6-0-0)
AI-192 - A library subprgram_body should replace, not complete, an instance (7-0-0)
AI-197 - Aggregate initialization of controlled subcomponents (7-0-0)
AI-199 - Does pragma convention for a generic unit apply to instances? (5-0-2)
AI-203 - S'Digits when T'Machine_Radix is 10 (6-0-1)
AI-205 - Priority Changes in abortable Part (6-0-0)

The following approved AI was not discussed and the **letter ballot** will be held soon:

AI-160 - Daylight savings and Ada.Calendar (10-1-2)

The intention for the following AIs was **approved but they require a rewrite**:

AI-130 - Should no_Local_Allocators disallow nested instantiations? (5-0-1)
AI-147 - Optimization of Controlled Types (6-1-0)
AI-161 - Default-initialized objects (7-0-0)
AI-188 - The definition of setting a task base priority is too vague (5-0-2)
AI-193 - Classwide Adjust and exceptions (5-0-2)
AI-195 - Streams (8-0-0)
AI-202 - Primitives of formal type derived from another formal type (no vote recorded)

The following AIs were discussed and **require rewriting** for further discussion or vote:

AI-085 - Questions about Append_File mode (12-0-0 -- old vote)
AI-131 - Interface to C - passing records as parameters of mode "in"
AI-162 - Anonymous allocators and tasks/finalization

The following AIs were discussed and a **first write-up assigned to an editor:**

AI-169 - Exceptions raised by Adjust/Finalize -- Missing case
AI-170 - Can an attribute defined in an annex be set in an attribute definition clause?
AI-198 - Pragma Convention(Intrinsic) is not a completion
AI-200 - Generic formal subprograms as dispatching operations
AI-204 - Interfaces.Fortran must be absent, right?
AI-207 - Pragma Inspection_Point affects only the current unit
AI-208 - What is the meaning of "same representation" in all partitions?
AI-209 - pragma Reviewable; can objects become uninitialized?
AI-210 - Questions on pragma Restriction No_Recursion and No_Reentrancy

The following AIs were not discussed **awaiting their pending rewrite or write-up**:

AI-051 - Size and Alignment Clauses for Objects
AI-058 - Accessibility Rules for Shared Passive Packages
AI-100 - Truncation required if Machine_Rounds false?
AI-109 - Size and Alignment Attributes for Subtypes
AI-117 - Calling Conventions
AI-133 - Controlling bit ordering
AI-148 - Requeue of protected entry calls
AI-153 - Picture String Grammar or Composition Rules Need Tightening
AI-158 - T'Class as generic actual type
AI-167 - Erroneous scalar Unchecked_Conversion?
AI-172 - Optional main subprogram ?
AI-173 - Optimizations and the use of 'Address
AI-174 - Are 'Read and 'Write guaranteed to be "inverses" for predefined types
AI-185 - Branch cuts of inverse trigonometric and hyperbolic functions
AI-186 - Range of root_integer
AI-187 - Task attribute operations are atomic but not sequential
AI-189 - The meaning of the terms "processor", "multiprocessor", and "processing node"

The following AIs were not discussed and still **seek an editor**:

AI-175 - Full conformance of expressions with attributes
AI-178 - Which I/O operations are potentially blocking?
AI-191 - An OBJECTive View
AI-196 - Assignment and tag-indeterminate calls with controlling results

Draft Conformance Testing Standard

The ARG did a paragraph by paragraph reading of the draft, beginning at section 5. Detailed suggestions for improvements were noted and Erhard will incorporate these comments in a revised version that will be distributed in the near future to the ARG. These minutes do not record these detailed comments, but rather summarize the general impression of the document.

On an editorial level, there are several almost verbatim repetitions. These will be eliminated. Some terminology was found to be inappropriate. At times, the terminology is not consistently applied (e.g., compiler vs. language processor). It was felt that, at several places, there was unneeded deviation from current practices or unneeded detail. The role of the ACAA in keeping commonality of the validation process seemed underdeveloped; for example, the ACAA seemed to have no say in the formulation of the testing procedures employed by the ACALs. There was surprise at the downplay of registration of derived compilers.

Language Enhancements

At the last meeting, WG9 passed a motion charging the ARG with also looking at language enhancements in view of future standard revisions. The ARG picked up this topic and collected a set of enhancement issues for further consideration, as time will allow. This list, in a very terse form of topics is:

- "circular types" (a.k.a. cyclic dependencies among specification units)
- Conventions of (overriding and primitive) subprograms (AI 117)
- Stream_Size attribute (AI 195)
- Pragma for preelaborable initialization of types (AI 161)
- "Unchecked_Union" for interfacing with C and C++
- Representation control for streams (e.g., endianness)
- specifying non-default bit-order (AI 133)
- allow overriding of fixed-pt. multiply (AI 163)
- user-defined dereferencing
- pragmas for specifying units (kg, m, inch, etc.)
- syntax to distinguish intentional overloading/redefinitions
- upward-closures for subprogram parameters
- support for mapping (Java) interfaces onto Ada
- additional annex for interfacing to C++ and Java
- "out" parameters for functions

AIs that closely relate to an issue are noted in parentheses. Excepting 163, these AIs are still "in the works" and, in almost all cases, a language enhancement would provide a much nicer and cleaner solution than a mere "interpretation" of the standard.

This is an initial list, more the result of a brainstorming than an agenda for future work, even though some of the issues on this list are very important to address.

At this meeting we discussed two of the enhancements not associated with any present AI: "circular types" and "Unchecked_Union". The minutes of these discussions are appended.

Next Meeting

The next meeting will be in the USA, most likely on the east coast. The dates of STC and of Easter will be avoided. Erhard will set a date and location shortly. On the east coast, Boston at Intermetrics or NYC at ACT are obvious choices.

Action Items

Action items were reviewed. Many have been accomplished. **Old action items** that remain on the list are:

Ben Brosgol: AI-153
Norm Cohen: AI-133, AI-167, AI-173
Gary Dismukes: AI-158
Bob Duff.: AI-051, AI-058, AI-109
Bob Eachus.: AI-100, AI-153, AI-172, AI-174, AI-185, AI-186 (2 meetings old!)
Pascal Leroy: extension AI to AI-161 (obsolete, see details on AI-161)
Stephen Michell: AI-148, AI-187, AI-189 (may have to be reassigned)
Erhard Ploedereder: AI-117 (was misassigned to Bob Eachus in the meeting summary)

Erhard and Bob: Prototype a few AIs to determine how to handle the remainder of the AIs in Standard Corrigendum form.(2 meetings old !)
Tucker Taft: Produce a draft AI on a language extension to allow for mutually dependent types in separate package specifications

Randy will assume Bob's role in drafting the Corrigendum.

New action items on the drafting or completion of AIs were assigned to the following ARG members:

John Barnes: AI-192, AI-204, AI-207
Randy Brukardt: AI-119, AI-184, AI-194, AI-197, AI-203, AI-205, AI-208
Gary Dismukes: AI-169, AI-200, AI-202
Pascal Leroy: AI-85, AI-131, AI-161, AI-168, AI-195
Erhard Ploedereder: AI-147, AI-164, AI-166, AI-190, AI-209
Jean-Pierre Rosen: AI-170
Tucker Taft: AI-130, AI-162, AI-188, AI-193, AI-198, AI-199, AI-210

The following action items were also assigned:

Erhard: Set the data and place of the next ARG meeting in the USA.
Erhard: Revise the Draft of the Conformance Standard.
Robert: Distribute an "Append" test to comp.lang.ada and collate results (AI-85).

Details on Enhancement Proposals

The first enhancement proposal deals with the issue of **cyclically dependent type definitions**, as they often arise when two or more graph structures contain nodes with components that reference nodes in the other graph structures, respectively. Within a single package, Ada solved the problem with the concept of incomplete types. Unfortunately, no good solution is available across the boundaries of packages. (The "Taft Amendment" of Ada83 applies only under very restricted circumstances.) The problem has reemerged in a different guise with the advent of object-oriented languages, most of which allow unrestricted mutual references between components of different classes. Mapping such a (frequently encountered) class design onto the Ada type model requires either a major rethinking of the data structures or the aggregation of the type definitions in a single package. Both approaches are highly undesirable and cumbersome.

Generally speaking, what is needed is a capability to extend the concept of incomplete types to apply across Ada packages. In deference to the Ada library model, an approach is preferred that preserves a linear compilation order despite the potentially cyclic nature of the type dependencies.

Tucker presented an initial design of such a capability.

Syntax:

with type [Ancestor_Name.]Package_Id.Type_Id
    [**is access** [**all** | **constant**] Subtype_Mark] ;
 package P is ....

In the first form of this context clause, the name Package_Id.Type_Id for an incomplete type is introduced. All the rules for incomplete types apply to this type in P, except that its completion is expected to be provided by the prefix package. However, no semantic dependency is created on that package; the package need not exist at all. Ancestor_Name, if present, must be the name of an existing package, which is implicitly "with"ed at this point.

In the second form, the name for an access type is introduced and its target type is made known. The target type may (but need not) be an incomplete type introduced earlier. The syntax allows for 'Class at this point to introduce a class-wide access type. All rules for access types apply.

In either case, discriminants are disallowed.

The compilation of a unit that has semantic dependencies BOTH on P and Package_Id must check that the named incomplete type does indeed exist in Package_Id. (Note that this is a compile-time, not a link time check !) This rule applies in particular to the body of P, if it withes the package Package_Id, as will usually be the case. Within this dependent unit, the type declarations in Package_Id act as completions for the incomplete types of P.

The model has some very nice properties: compile-time checkable, no disturbance of the linear compilation order, linking of a system without package Package_Id is possible and meaningful, since, in that case, the type remained incomplete and cannot have any effect on the execution.

Why the need for identifying a target type ? Because some ILs require that access types identify their target type (e.g., Java bytecode).

During the discussion, the question was raised, whether it wouldn't be possible to add additional information to the incomplete type, similar to the contract promises made by generic formal types. This was not pursued in detail, as it seemed to complicate the approach without gaining much. It would affect only usage of the type in the package specification, since the body can always refer to the full information simply by withing the respective package.

Another question was whether child units might provide a means for introducing incomplete types in a parent and completing them in a child. At first and second glance this seemed to introduce many problems that the above approach avoids, in particular problems that are caused by the exporting of incomplete types. Note that the proposal above does not make the incomplete type visible to the outside world.

As an issue with the proposal, an interaction with the "remote types" classification was noted, which needs to be resolved.

Tucker will write up the proposal in an AI.

The second enhancement discussed was the issue of modeling C unions in Ada when interfacing with C and C++. Tucker presented his proposal for this capability, termed **"pragma Unchecked_Union",** and contrasted it with the approach taken by GNAT. The main difference between the two approaches can be characterized as follows: The GNAT model maps a union to a discriminated record, with the alternatives of the union being single component variants of the single variant part of the record. The discriminant is not represented in the record layout. A consequence of this mapping is that anonymous unions, used quite often within C structs, need to be mapped as a named record type in Ada and the union in the struct modelled as a single component. The Taft model maps anonymous unions roughly to a variant part and thus is able to "flatten" out C structs containing anonymous unions of structs into single Ada record types.

We mainly discussed the following questions and issues in the Taft model to explore the technical consequences and any difficulties in answering them affirmatively. In all cases, the simpler but much less elegant GNAT model denies these capabilities, while the Taft model appears to allow them.

1) Should we allow a non-variant part?
2) Should we allow a list of components in a variant alternative?

These two questions are at the heart of the difference between the two models. Technically, there appears to be no reason to disallow them.

3) Should we allow nested variants?
4) Should we allow multiple discriminants?

These two questions address the degree to which various C structs can be mapped to a single Ada record type. Again, no technical obstacles were found.

5) Should we allow defaults on non-discriminant components?

The answer to this question is arguable, as the C code will clearly not obey such defaults and hence provide no guarantee for validity of the component values. However, in mixing languages such guarantees cannot be provided anyway, regardless of the specification of default values in one language. One might still provide the convenience of defaults for objects created by Ada code.

6) Should we not require defaults on discriminants?

There is a slew of issues regarding the size of such record objects and the answer to this question will be heavily dependent on the size prescriptions.

7) Should we allow the declaration of constrained subtypes?

There doesn't seem any technical harm in doing so (as long as the constraint is not used to calculate size for a non-limited type). Of course, no constraint checks are possible. However, if the constraints are static, compilers might be nice enough to issue stern warnings against constraint violations.

Related questions were:

1) What size should be allocated for objects?

The appropriate answer is to allocate the maximum size if the type is non-limited (to accomodate assignments by the C code without memory implications), while the constrained size may be sufficient when the type is limited. This of course presumes that the C code adheres by contract to the limitedness declared for the Ada type or else serious damage can be done by memory overwrites.

2) What should equality and assignment mean?

We would like them to mean a block compare and block move. These semantics are safe for those types with a maximum size allocation.

One observation made was that, quite obviously, discriminants of these record types are not allowed to be used in any other context but the expression of a variant part (e.g., no discriminant-dependent array bounds).

Details of the AI Review

AI-*

It was observed that many authors did not update the administrative data of the AIs (version numbers, dates on the sections, etc.). Randy is asked to check specifically for the accuracy of these data before entering the AIs in the data-base. **Authors are asked to pay more attention to these data**.

Many binding interpretations come in with empty !wording sections. The !wording sections should be filled and should at least identify the affected paragraphs, e.g. "Paragraphs x.x(x) and y.y(y) need to be adjusted to reflect the intent of the AI." "see summary" can be used, if the summary identifies the paragraphs.

AI-85

Initial sentiment was that the third paragraph of the !summary should be changed to give an unconditional permission for the Use_Error alternative, to let it be applied when positioning support is available, but unsuitable for the postulated semantics. The AI should then be turned into a binding interpretation (with wording section).

Further discussion then asked the question for the prevalent implementation behavior. Robert volunteered to write a test program, eliciting the "C/Unix model", "the Ada model", the "Use_Error model" and anything else, distribute it to comp.lang.ada, tabulate the results and pass them on to the ARG.

The AI will then be adjusted by Pascal depending on the outcome of this experiment. If the outcome demonstrates that most implementations follow the Ada model, the AI will be worded as implementation permission to raise Use_Error. Otherwise, Use_Error will be the required behavior.

Depending on the amount of change and the divergence from the initial consensus above, the AI may then go out for editorial review rather than further discussion at a meeting.

AI-119

Apart from the necessary updates to the status information, no further changes are needed. The old approval vote stands.

Randy will do the final update.

AI-130

We discussed the issue of data flow dependency, in particular in the context of conditional compilation. (if Debug then...). The consensus is that the "primary" semantics should be expressed as a static compilation and/or post-compilation check, but that an implementation permission should exist that allowed the omission of the check in code that is unreachable and actually not generated by the compiler.
The first sentence of the recommendation should go to the summary to make clear that post-compilation enforcement is allowed. The !wording should be filled. In the subject and the !summary, several members felt that "may" should be replaced by "might".

The intent of the AI was approved 5-0-1. Tucker will update the AI.

AI-131

There is agreement with the basic content. Partially in light of the discussion on enhancements in general, partially in response to the WG-9 motion, the proposed pragma will become a mandatory part of the Annex as opposed to a recommended implementation-defined pragma.

Both the verbiage and the details on the semantics of the pragma need to be refined. It should roughly read as "...that any 'in' parameter of the type T is passed by copy to a subprogram of convention C. The convention of the type is implicitly set to C. The pragma is applicable only to (untagged) record types. "
The last paragraph of the !response is deleted.

Pascal will very soon rewrite the AI accordingly. Erhard would like to bring this AI to WG9 in November.

AI-147

Tucker raised the issue of external effects within the omitted calls (as opposed to after the calls), which aren't yet covered by the AI. A change to accommodate them is to alter the permission in the !summary to refer to "additional effects" rather than only "side-effects on other objects".
Jean-Pierre then explored the question whether every object created will be finalized. Erhard discovered that the wording of the !summary (unintentionally) did not cover the case, where an explicit initialization by aggregate is done (which, by another AI, must be done directly in the object without Initialize or Adjust calls). While this can be fixed by adding the case in the AI, the group went on to revisit the decision made at Burlington to exclude Initialize from the optimization. The opinion swayed towards the Henley model (i.e., not to distinguish initialization and assignment), in order to go for the simple rule: For limited controlled types, Initialization and Finalization are guaranteed to occur, while for controlled types only the intermediate usage of the object's value between Initialize/Adjust and subsequent Finalize will prevent the option of optimizing away the assignment operation along with the corresponding implicit call and subsequent Finalize.

This Henley intent is approved 6-1-0. Erhard is asked to "revert" the AI accordingly.

AI-161

This is a binding interpretation, not a confirmation. The order of the !summary paragraphs should be inverted. The question should include 10.2.1(5) to make sense. The wording should use "does not" in lieu of "doesn't". In the !discussion, change "the.." to "a full view". The second paragraph of the wording needs to be changed into appropriate bullets under C.4.(4) to exclude objects of controlled type, descendants of generic formal types, private types and private extensions. We discussed as some length, what type features are likely to cause code to be generated.

On a deeper issue, the question was (re-)raised why the pragma was not introduced in this AI, and how much its later introduction would alter the semantic content of the AI. Given the WG9 attitude that the ARG can begin thinking about extensions, here would be a quite perfect opportunity to "do the right thing". It was also suggested, but not discussed in detail, to let the pragma apply to all types, not just to the types that cause the problem. A further suggestion was to explore using such a pragma as a means to assert properties of generic formal types to be enforced at instantiation time.
Consensus developed to include the pragma directly in this AI and not write a separate "extension" AI, especially if this helped spelling out a simpler solution.

The table of types affected by the issue should drop all those, whose package is not preelaborable. Ada.Strings.Maps.Character_Mapping, Ada.Strings.Maps.Character_Set and Ada.Strings.Unbounded.Unbounded_String change from "no" to "yes".

The above intent on adding a pragma was approved 7-0-0. Pascal will make the changes.

AI-162

It was observed that the terminology used ("executing a master") inadvertently led to a semantics that allowed altogether omitting execution of the constructs in question, which is clearly wrong. It was observed that the AI first goes to great trouble of defining new masters and then gives permission to implementation to ignore them altogether. It was then mused whether one needs to use the heavy gun of changing the master concept to come to grips with this issue. The group was clearly divided on this issue. In any case, both the finalization of controlled objects and the completion semantics of task objects need to be handled. The various alternatives presented in the discussion were discussed again in search for new insights.
Straw votes establish that 7-0 can live with a change to the master concept. 3 favor the master approach, 4 prefer a special "deferred waiting" rule. It is clear that the issue needs to be worked some more, either in the direction of a

simpler "master"-extension, or special rules.

The AI was then tabled without further resolution.

On the second day, consensus (7-0-0) was established to a) simplify the AI significantly, b) tie into 7.6.1(13) and 3.10.2(13) and include impl. permissions there.

The title of the AI needs to be fixed, e.g., "Anonymous access types and their masters".

Tucker will take a look into simplifying the presentation.

AI-164

Several editorial fixes were requested: change "e." to "E." in the !question. Add the question why the target type needs to be private. Correct the hyphenation of RACW. Delete reference to the Henley meeting in the wording section. Change the 1.para. of the !wording to better terminology to express the recursive nature of the private extensions.

The AI was approved (7-0-0) with these editorial changes. Erhard will do the edits.

AI-166

Approved with minor editorial changes (7-0-0). Assigned to Erhard.

AI-168

The summary should be expressed in terms of "is illegal" to assert the compiler's obligations. The title should add a "not". Approved with these changes (7-0-0). Pascal will make the change.

AI-170

Annex attributes cannot be set unless stated otherwise in the respective Annex.

Jean-Pierre will write-up the AI.

AI-184

In the !summary, fix the typo "object" -> "object's"

Approved with this editorial change (7-0-0). Randy will make the change.

AI-188

After some discussion, consensus developed to confirm the RM rather than the intent of the comment for reasons explained in part in Ted's rejoinder.

The reversal is approved (5-0-2). Tucker will do the rewrite.

AI-190

The question of this AI should be rewritten to be of the lesser scope that is present in the comments and the summary. In the summary, change the verbiage to "of a pragma Restrictions" and delete 2. paragraph. Add a reference to D.7(15). Change the title to "Compile-time enforcement of pragma Restrictions". Make it a binding interpretation. The wording should refer to 13.12 and might reincarnate AARM 13.12(9b).

Approved with these changes (6-0-0). Erhard will make the changes.

AI-192

Approved (7-0-0) with a change to replace the term "noninstance" in the !wording by something more appropriate. John will make the change.

AI-193

Discussion revealed a misunderstanding of the term "initialized by assignment", which was meant by Tucker as explicit initialization by other than an aggregate. This needs to be clarified in the AI. !wording needs to be filled in.

Discussion then turned to intent. It was observed that the comment actually asked for a relaxation, while the AI is currently more in the spirit of tying down semantics into one direction, rather than giving more freedom to implementations to exploit their idiosyncrasies in this bounded error situation. Two implementation models with contrary semantics actually in use were discussed and there did not seem to be a possible compromise to accommodate both without giving up on being specific on the semantics. There also are reasonable arguments for and against Finalization after an Adjust with exception.

Thus, the consensus was to leave it unspecified whether Finalize is called after Adjust raised an exception.

This intent was approved (5-0-2). Tucker will do the rewrite.

AI-194

This looks like a typographical error. Randy will investigate and, if so, add it to AI-presentation.

AI-195

We decided to postpone the discussion on stream base range to the second day, but to examine the other issues presented by the AI.

In the 1.para., the word "default" should be added before "initialization", in the 2. para. "[limited] type", in the 3. para., "limited [tagged]". Early in the !discussion, "takes [place]".

There was a lengthy discussion over current component-wise semantics of the calls on Read/Write of a stream. What is desired is added freedom to let predefined 'Read and 'Write of composite types to act in larger chunks, when possible, but guarantee that there is no buffering at the attribute subprogram level. The proposed rule requiring at least one call seems to run afoul, however, of null arrays and records. Pascal will fix this somehow.
Discussion of the other questions and conclusions of the AI did not yield any requests for changes.

The AI was then tabled for the 2. day.

On the 3. day we examined the stream representation. A satisfactory solution is to define the default stream size of elementary types as follows:

- for floating-point and decimal types this size is implementation-defined
- for all other elementary types:
    - if the minimal size required is less than the word size, then the stream size is the minimal size increased to the smallest integral factor of the word size,
    - if the minimal size is larger than the word size, the stream size is the minimal size increased to the smallest integral multiple of the word size

The intent was approved (8-0-0). Pascal will complete the write-up.

AI-197

The AI-title was changed to "Aggregate initialization of controlled subcomponents".
It should have a !reference to AI-83.

Approved with these changes (7-0-0). Randy will fix the AI.

AI-199

After some discussion consensus developed for the conclusion of the AI. An addition to the AI is requested that introduces the notion that a pragma explicitly given for an instance will override a pragma "inherited" from the generic unit.

Approved with this addition (5-0-2). Tucker will add the necessary words.

AI-202

The AI was briefly discussed. It was felt that the issue was difficult to grasp. Several sentences were suspected to have one indirection too few. Gary was asked to try to come up with a summary and recommendation that would be easier to understand. As it was late in the day, the AI was tabled for discussion on the next day. It was not picked up again for the rest of the meeting.

Gary will take another look at the AI.

AI-203

In the !question, add "no" as answer to the question. Add "see summary" to the !wording section. The title should be changed to something more appropriate.

Approved with this change (6-0-1). Randy will fix the AI.

AI-204

The consensus is that the package must be absent if the convention pragma is not meaningfully supported (e.g., for Fortran, multi-dim. arrays better have column-major representation). In order to be meaningfully supported, the compiler provider must identify the compiler(s), with which the conventions agree. For COBOL, the compiler need not be on or for the same host of the Ada compiler; this allows for the fact that the interfacing capability is mainly provided to provide a means for binary compatibility of file contents.

This interpretation applies to conventions and interface packages of all languages.

John will write the AI.

AI-205

Change "that it must be" to "that the update must occur" in the !question. Check punctuation.

Approved with these minor editorial changes (6-0-0). Randy will do the updates.

AI-206

Approved without change (7-0-0).

AI-207

This will be a binding interpretation. The pragma without arguments applies only to all object whose declarations are visible at the inspection point. Anything more encompassing is technically impossible in a separate compilation model. However, it is safe to assume that a compiler supporting Annex H has a global setting that guarantees that all global variables are up-to-date at any execution point, anyway.

John will write it up.

AI-208

The paragraph is much too restrictive. It would be outright foolish to mandate it in general on a heterogeneous system without shared memory. If there were shared memory (e.g., on an embedded system board), the compiler would have to be much more clever about it anyway; consider, for example, incompatible floating-point representations.

The consensus is to delete the paragraph in its entirety, since on a homogeneous system the compiler can be trusted to do this anyway.

Randy will write it up.

AI-209

In H.3(9), the term "reference to" definitely needs to be replaced by "reading of" or similar usage terminology.

Erhard and Robert had a lively discussion, whether or not "uninitialized" should be a transitively "infecting" concept, as would be true for the "invalid" property, and how much flow information needs to be used in the analysis.

Erhard will write up the AI and pass it to the HRG for comment.

AI-210

The answers are "no" to the first question and "yes" to the second one.

Tucker will write it up.