Final Minutes of the 3. ARG Meeting
Bennington, VT, USA, 7-9 October 1996

Attendance: Barnes, Cohen, Dewar, Dismukes, Duff, Eachus, Ishihata, Kamrad, Leroy, Michell, Ploedereder, Schonberg (for days 1 & 2).

Host facilities were graciously supplied by Robert Dewar and ACT. The ARG expressed its appreciation.

These minutes first summarize the voting results on the AIs that were discussed, then present all procedural issues that came up during the meeting and the action items assigned. The record of the detailed discussions on AIs is appended in AI order for easier reference. This order does not correspond to the chronological order in which the AIs were discussed.

Summary of the action taken on the AIs:

Approved in final form (some with minor editorial changes):

AI95-00004/02 -- Conversions to types derived from remote access types
Approved (12-0-0)
AI95-00031/00 -- Unpacking a record type with primitive subprograms
Approved (11-0-0)
AI95-00035/01 -- Type descriptors can be laid out at compile time
Approved (12-0-0)
AI95-00037/04 -- In Interfaces.C, nul and wide_nul represent zero
Approved (12-0-0)
AI95-00040/02 -- Does <> for a formal subprogram default freeze the actual ?
Approved (11-0-1) with change to title
AI95-00041/04 -- Program unit pragmas in generic units
Approved (12-0-0)
AI95-00044/01 -- Overriding of Declarations
Approved (11-1-0) with change to title
AI95-00048/04 -- Can an RCI unit be a library subprogram?
Approved (12-0-0) with change to title
AI95-00071/01 -- Correction to the Valid function in COBOL Interface
Approved (8-0-4)
AI95-00072/01 -- Clarification of result length for conversions in COBOL Interface
Approved (7-0-5)
AI95-00087/00 -- Saving and restoring Current_Output
Approved (12-0-0)
AI95-00089/01 -- Float_Random.Value, Discrete_Random.Value
Approved (11-0-0)
AI95-00106/02 -- Freezing Rules
Approved (11-0-0)
AI95-00107/03 -- Base attribute for non-scalar subtypes?
Approved (12-0-0)

AI95-00110/01 -- No Constraint Check on 'out' Parameter of an Access Type
Approved (12-0-0)
AI95-00112/01 -- Wide_String file names?
Approved (12-0-0) with change to title
AI95-00118/01 -- Termination signals query of Terminate attribute
Approved (9-0-3)
AI95-00127/01 -- Expected type of a 'Access attribute
Approved (8-0-4)
AI95-00128/01 -- String Packages
Approved (12-0-0)
AI95-00136/02 -- Placement of Program Unit Pragmas in Generic Packages
Approved (12-0-0)

Approved in final form, but to be editorially reviewed, since this was the first vote on these confirmation AIs:

AI95-00006/00 -- private child in with_clause
Approved (10-0-0)
AI95-00012/00 -- The first subtype of a type defined by an access[_type]_definition
Approved (10-0-0)
AI95-00018/02 -- Inconsistency with Ada'83 in the definition of exponentiation
Approved (10-0-0)
AI95-00025/01 -- Matching rules for generic formal access-to-constant types
Approved (9-0-1)
AI95-00032/00 -- visible part of a renaming
Approved (10-0-0)
AI95-00042/01 -- use of incomplete types
Approved (10-0-0)
AI95-00045/01 -- Entry calls
Approved (10-0-0) with a change of title
AI95-00077/01 -- Separate compilation of generic bodies
Approved (6-1-2)
AI95-00078/01 -- semantic dependence on illegal or obsolete units
Approved (11-0-0)

Approved (with changes) for editorial review:

AI95-00033/03 -- delayed declaration of inherited primitive subprograms
Approved (11-0-0) with editorial review
AI95-00034/01 -- Unconstrained Formal types
Approved (9-0-2) with editorial review
AI95-00095/01 -- Modular types on one's complement machines.
Approved (6-0-5)
AI95-00097/04 -- Conversions between access types with different representations.
Approved (10-0-2)
AI95-00121/01 -- Pragma Attach_Handler on Nested Objects
Approved (11-0-1) with editorial review

AI95-00123/01 -- Equality for Composite Types
Approved (10-0-2) with editorial review
AI95-00124/00 -- Ligatures Are Allowed in Identifiers
Approved (11-0-0)
AI95-00145/01 -- Profile of predefined operators for scalar types
Approved (7-0-3)

Approved pending letter ballot:

AI95-00126/03 -- Remote_Types Packages
Approved (8-0-2) letter ballot requested

Discussed but sent back for substantive revision:

AI95-00051/03 -- Size and Alignment Clauses for Objects
tabled
AI95-00058/01 -- Accessibility Rules for Shared_Passive Packages
tabled
AI95-00064/03 -- Elaboration checks for renamings-as-body
tabled
AI95-00085/02 -- Questions about Append_File mode
tabled
AI95-00104/01 -- Version and Body_Version attributes
Approved (12-0-0) with editorial review and Annex E author feedback
AI95-00109/02 -- Size and Alignment Attributes for Subtypes
tabled (jointly with AI-51)
AI95-00117/01 -- Calling Conventions
tabled
AI95-00131/01 -- Interface to C -- passing records as parameters of mode 'in'
rewrite (7-0-5)
AI-95-00132/01 -- Exceptions Raised at End of Stream
Intent approved (10-0-1) for alternative 3
AI95-00141/03 -- Exceptions in Interfaces.C and its Children
Rejected (2-5-3)
AI95-00147/01 -- Optimization of Controlled Types
Intent Approved (5-1-4), see minutes
AI95-00157/01 -- Visibility of Inherited Private Components
Intent Approved (12-0-0) for editorial review
AI95-00158/00 -- T'Class as generic actual type
tabled
AI95-00163/00 -- User-defined fixed-fixed multiplying op
Intent established (10-0-1), see minutes

Deleted:

AI95-00155/01 -- Stream-Oriented attributes for language-defined private types

Procedural Issues:

Erhard reported contacting the AdaIC about HTML-versions of approved AIs to be made available on sw-eng to the general public. The AdaIC is hard at work to make this happen.

On the question of producing standard Corrigenda, Erhard reported that he has asked for but not yet received editorial instructions from Bob Mathis.

How to handle editorial changes to AIs that have already undergone editorial review and how to handle rejection of editorial comments by the AI author ? To avoid polishing AIs to death, we adopted the procedure that unless the commentor demands a vote on the content of their rejected comment, no additional vote will be taken. Similarly, such AIs with additional editorial changes decided at an ARG meeting will not be delayed by yet another editorial review.

There was general consensus that the requirement for new Wording in AIs will continue to be considered on a case-by-case basis and it was suggested that missing Wording should be supplied by the requestor of such wording.

The need to vote on simple confirmations was raised again. It was decided to deal with confirmation AIs that already have a number in the standard fashion, while new simple confirmations should go into one big "Confirmations" AI.

There is an apparent problem in the current ada-comment setup, which forwards to ada9x-mrt@inmet.inmet.com. If the Intermetrics system "coughs", redistribution of ada-comments gets delayed and there is the danger of loss of comments. Action Item: Erhard will contact Tuck about the state of the Intermetrics software to determine how the comment data base plus software could be moved to sw-eng, then to contact AdaIC to solicit their support of the system.

There was some discussion about an additional meeting per year for the purpose of speeding up the AI examination and approval process. However, most ARG members have funding problems that makes it difficult to travel to an extra meeting. Besides, the deliberate process seems to have kept pace with the current flow of AIs and has, in some cases, prevented mistakes of premature AI resolutions.

Next Meeting:

John Barnes offered to host the next meeting in the UK, Greater London area, from April 11 to 13, 1997, adjacent to the RT-Workshop in Wales. The dates may change by a day, depending on whether the organizers of the RT-Workshop are willing to move their meeting by a day off the 13th. This offer is gratefully accepted by the ARG and the venue and time of the next meeting are so decided.

Action Items:

Pending old action items:

Dewar: AI-54
[Landwehr: analyze old UIs; post remaining ones to ada-comment]
Ploedereder: get corrigenda format from Bob Mathis
Duff: create subdirectory of WG-9 approved AIs

New action items:

Taft: AI-158
Annex E experts: AI-58, AI-104
Barnes: organize next meeting
Duff: post editorially revised AIs, AI-*, split stream AI from AI-85
Ploedereder: minutes, prepare for next meeting, check possibility of transfering ada-comment data base to sw-eng, invite Gnat distribution team to London meeting, notify ACVC team about AI-127 and AI-128; present approved AIs to WG-9
Kamrad: minutes
all: editorial review of approved AIs as soon as posted

==========

Details of the AI Review:

AI-04 -- Conversions to types derived from remote access types

The ballot brought out comment(s) on the intent of the AI. Storage pools are perceived to be in the partition in which referenced objects are allocated. Remote access types should NOT be permitted to do allocation and this restriction is the premise of the AI, cast in terms of storage pools. Robert Dewar questioned this rationale; he opines that the storage pool can be distributed (implying the use of "fat pointers"). But the conceptual memory model that supports primarily a client-server model and avoids implicit communication calls precludes such a distributed memory model. Robert conceded that the latter model is the one really intended by the language. More text in the Distributed System Annex explaining this model would have been beneficial. Approved as is (12-0-0).

AI-06 -- private child in with_clause

Approved without substantive discussion (10-0-0).

AI-12 -- The first subtype of a type defined by an access[_type]_definition

Approved without substantive discussion (10-0-0).

AI-18 -- Inconsistency with Ada'83 in the definition of exponentiation

Approved without substantive discussion (10-0-0).

AI-25 -- Matching rules for generic formal access-to-constant types

Approved without substantive discussion (9-0-1).

AI-31 -- Unpacking a record type with primitive subprograms

Editorial changes requested by reviewers had been suitably reflected.
Approved 11-0-0.

AI-32 -- visible part of a renaming

Approved without substantive discussion (10-0-0).

AI-33 -- delayed declaration of inherited primitive subprograms

There are two problems here, if the RM stays as is. First, without adding the "immediately within the declarative region" rule, the implicit declaration can appear in nested scopes which is surprising. Second, without this rule, there is an upward incompatibility with Ada83 and thus a very compelling reason to correct the RM.

It was suggested to change the example by adding a call to Op (value of type T), which explicitly shows the illegality. Stephen raised the question about the relationship of this AI to the issues addressed in AI 157. Since there is a similarity, it was recommended to add a !reference to AI-157 which is also a ramification of visibility rules.

Approved 11-0-0 with these editorial changes.

AI-34 -- Unconstrained Formal types

Erhard was concerned that ignoring the constraint on the instantiating type might open a hole for out-of-range indexing. He provided an example...[that was unfortunately lost, but is recreated in spirit as follows]

```
generic
  type ptr is access all string;
  O: in out ptr;
package P ...
package body P is
  subtype str_ptr is ptr(26..28);
  X: aliased string(26..28);
  Y: str_ptr := X'access;
```

```
  O := Y;   ...
end P;

type ptr1 is access all string;
subtype string20 is ptr1(1..20);
Obj: string20;
package New_P is P(string20, Obj);
```

Somwhat surprisingly, the assignment of "O := Y" must yield a run-time check of the bounds of the pointer subtypes of the two variables. This check must fail for this instantiation. (If that were not the case, havoc would result for indexing into Obj after the instantiation.)

This AI points out an unfortunate consequence but it is necessary to remain upward compatible with Ada83. It was approved 9-0-2 subject to editorial review.

AI-35 -- Type descriptors can be laid out at compile time

Editorial changes requested by reviewers had been suitably reflected.
Approved 12-0-0.

AI-37 -- In Interfaces.C, nul and wide_nul represent zero

The editorial changes were limited to the presentation in the Question section where the intent is explained in the two paragraphs with the "yes" answers. After some spirited discussion on the technical issue, the group arrived at the same technical conclusion.
Approved 12-0-0.

AI-40 -- Does <> for a formal subprogram default freeze the actual ?

Editorial changes requested by reviewers had been suitably reflected. The subject line is now stated in the affirmative.
Approved 11-0-1.

AI-41 -- Program unit pragmas in generic units

The changes were limited to fixes of typos and a sentence added to the Summary, stating the rule for pragma Inline. The Appendix of this AI was inadvertently left off and will be added.
Approved 12-0-0.

AI-42 -- use of incomplete types

Approved without substantive discussion (10-0-0).

AI-44 -- Overriding of Implicit Declarations

The Wording section is missing which caused ARG members to withhold approval during the letter ballot. The opposing view is that the ARG should not always spend the time to find the exactly right wording, since this is difficult, time-consuming, and incurs the risk of making mistakes with the new wording. The Summary seems to have enough information to guide implementers. The title was changed. Approved 11-1-0.

AI-45 -- Entry calls

It was decided that the subject of the AI should be changed to something less encompassing. Otherwise approved without substantive discussion (10-0-0).

AI-48 -- Can an RCI unit be a library subprogram?

This AI reversed the intent of its original version and was passed 11-0-1 in the letter ballot. An editorial change of turning the subject question into an affirmative statement was agreed upon. Approved 12-0-0.

AI-51 -- Size and Alignment Clauses for Objects

Pascal recommends that parts of the Discussion be moved to the Wording section.

Robert Dewar calls for an explanation of the Summary, revisiting the previous technical discussion, specifically on point 2 of the Summary. He believes that this point overspecifies the convention with the "and should not" phrases. He argues that there are processors, like the x86 (and R6000) where alignment differences are not significant and therefore coercing an implementation to make it significant (with the wording, "and should not") is bad practice and misleading. It also runs counter to the additional implementation flexibility that has been introduced into Section 13. Stephen and Erhard are unconvinced by Robert's argument and argue portability and misguided procurement eagerness. Robert reenforced his point by stating that the term "need not" permits an implementation to do what it can correctly and to reject that which it can't do correctly. Erhard conceded on the alignment issue, but holds fast to the size issue. He provides an example to describe the technical issue:

```
type my_int is range 0..255;
for my_int'size use 32;
type my_int_acc is access all my_int;
...
type R is record
    x:  aliased my_int -- at 0..7;
    y:  aliased my_int -- at 8..15;...
```

Any reasonable implementation (without pointers fattened specifically for that purpose with size information) will simply have to reject such a component clause for x. Creating correct code for a dereferenced my_int_acc yielding x would indeed require fat pointers, which we surely should discourage.

A straw vote to remove "should not" is approved 9-3-0.

Next the discussion turned to consider Robert Eachus' editorial comments.

Robert Dewar complains that the no-maximum-size rule for composite objects will require implementation to not take advantage of efficient default implementations. He recommends that composite objects be handled like fixed point, floating point and access types as described by the previous point of the AI, namely, it need not support a size that would not be chosen by default.

Robert Dewar brings up an example to highlight some of the composite type/object sizing

```
type q1 is record
  x:  integer;
  y:  string (1..ident_int (2));
end record;

type q2 is record
  x:  integer;
  y:  string (1..2);
end record;
```

The DEC implementation insists on support of size-clauses that force both records to have a size of 48, but ACT has sizes of 64 for the first and 48 for the second. Ed Schonberg points out that RM 13.1(23) says that support of a size clause is not required for the first case. Erhard notes that this is an interesting example, but that it is beside the point, since the AI does not deal with the smallest possible size, but rather with the largest size to support.

It is also pointed out that the wording doesn't cover the sizing and alignment implied by component clauses, which is a very tough issue to implementations.

A vote on the AI was postponed until Tuesday, when it was agreed to table the AI until the discussion has been reflected in the AI.

AI-58 -- Accessibility Rules for Shared_Passive Packages

The discussion focused on the visibility and lifetimes of Shared Passive packages as it pertains to the access to and from packages in its own partition and other partitions. A set of rules to get correct accessibility without unduly expensive checks for violations are intended in the RM as follows:

- Shared passive partitions live at least as long as active partitions
- Active partitions access passive partitions but never the reverse
- Semantic dependence with a "with" clause permits the withing package not only visibility but reliable access

The examples from the discussion section of the AI illustrate these rules:

- Example 1 shows how a compile check prevents passive access to a package it doesn't with.
- Example 2 shows how a compile check prevents access by a passive partition to a shared passive package it doesn't with.
- Example 3 shows that a compile check cannot prevent access, at runtime, by a shared passive package to a package to which it doesn't have visibility. A simple runtime check (simple comparison of integer values assigned to packages) is proposed to address straightforward access (through a type conversion and through the passing of access parameters) as illustrated by the example. More complex access checks are not known and thus these accesses can be erroneous.
- Example 4 addresses several additional points. The first half of the example shows how the equal accessibility levels of shared passive packages prevents legal accessibility of a body of one of those packages to the spec of the other package despite the presence of proper withing between the body and spec.

Then the discussion turned towards methods to permit co-location/co-addressing of mutually dependent shared passive packages (from a system viewpoint), as illustrated by example 2. The two choices appear to be: either to use runtime checks to find out if both passive packages have correct accessibility or to introduce a pragma to put those packages into partitions/hardware configuration that permit correct accessibility. The pragma alternative was the preferred choice and the discussion turned towards proposals that were acceptable and tasteful.

Two typos were noted: "than [,] that" in the summary and "{F1.}T1" in package body P2.

The AI was then tabled, expecting an in-depth review by Annex E specialists.

AI-64 -- Elaboration checks for renamings-as-body

Erhard had trouble with the current Wording section, since the term "execution of the body" is overloaded to mean both the elaboration of the body (declarative_item) and the execution of the body as part of calling the subprogram. There doesn't seem to be any need for this potentially confusing reuse. Besides, he notes that the summary doesn't say for which subprogram the checks are done and when. The Summary should be expanded to state that elaboration checks need to be performed during the call and that two checks are being done (namely, that the renaming and the renamed subprogram body are elaborated.). After much discussion the AI was tabled until Bob can modify the AI to address these concerns.

AI-71 -- Correction to the Valid function in COBOL Interface

This AI fixes a bug in the interface description. Both Robert Dewar and Robert Eachus verify that the AI proposes the correct solution. Approved 8-0-4.

AI-72 -- Clarification of result length for conversions in COBOL Interface

Return strings always have a lower bound of 1 to match COBOL conventions. Tying the length to Format as recommended by the AI fits COBOL methods better than tying the length to the minimal number of characters. The AI is doing the right job. Approved 7-0-5.

AI-77 -- Separate compilation of generic bodies

Compiling here means checking for the legality of the instantiation, not necessarily the actual creation of code or another recompilation.

The details of the Note in 10.1.4(10) (which is the subject of the AI) were discussed. In particular, the Note says in effect that changing the generic body must not force the recompilation of all client instantiations. Obviously, code for the instantiated bodies needs to exist no later than (successful) linking. An implementation is free to force steps to cause the (automatic) code generation for all the instantiations of the generic unit (among other steps it could take) before linking.

The point of this Note was a specific user requirement during the revision process and nothing to the contrary has occurred to change this requirement.

Gary Dismukes pointed out that the Summary was not directly responding to the original question. But apparently not worth making changes.

Approved as is (6-1-2).

AI-78 -- Semantic dependence on illegal or obsolete units

Robert Dewar points out the ASIS group has spent lots of time confirming the point of the last sentence of the Summary.

Approved (with dropping of "ensuring" in the last sentence of Summary) 11-0-0.

AI-85 -- Questions about Append_File mode

This AI has as yet reached no conclusions with regard to operations in Append Mode. The discussion notes that there also are problems with the lack of an IN_OUT mode for Stream_IO.

Robert Dewar argues that Ada-unique I/O is a bad decision and we should match C or Unix conventions for I/O. Unfortunately it may be too late to make these corrections.

References to A.12.1(28, 35) and A.8.2(16) need to be added.

The AI appears to answer the append-specific questions with respect to Tuck's response but it says nothing substantial about the stream problem. Consequently this AI will only address the

specific questions regarding Append mode and Set_Index as raised by Keith Thompson. A new AI will need to be created to handle the other identified problems of Stream_IO.

AI-87 -- Saving and restoring Current_Output

The only change was removing a Scribe command. Initially Erhard decided no vote was needed due to the insignificance of the change but later returned to a vote nevertheless in order to treat all AIs consistently. Approved 12-0-0.

AI-89 -- Float_Random.Value, Discrete_Random.Value

Recent changes by Bob Duff addressed the comments from Pascal Leroy, Robert Eachus and Jean-Pierre Rosen and were confirmed by the meeting. Approved 11-0-0.

AI-95 -- Modular types on one's complement machines.

A request has been made to make an exemption for modular types on one's complement machines. These architectures are disappearing and so is it worth the work to grant the exemption by explicit binding interpretation? An easier alternative will be to encourage the implementer to use the permission granted by the old Ada83 AI-325.

Three choices were enumerated:
1. Do nothing
2. Allow for the "hole" in the range of supported values or allow an extra supported modulus value outside the base range.
3. Tell the implementer to use a non-standard type to achieve his goal.

The third alternative was felt to be too restrictive because of the rule prohibiting such types as formal types in generics.

The meeting chose the current approach taken in the AI by 6-0-5.

Discussion returned to the rule that nonstandard numeric types are not permitted as actual parameters to generics. Robert Dewar argued that this limitation should be implementation-defined rather than being imposed on all such types. Instead of solving this problem, the meeting chose to wait until someone complains about it to ada-comment.

AI-97 -- Conversions between access types with different representations.

The only requested change was to add the parenthetical remark in the Summary.

Stephen Michell wants the type conversion to be implementation-defined, revisiting the same arguments from the last meeting. Robert Eachus proposed that "implementation-defined" wording should be added to the Summary. Eventually it was agreed that the words "not specified" in the Summary and "unspecified" in the 1. line of the Response be changed to "implementation-defined". A straw-vote on this change yielded 7-2-3.

Next came a discussion on whether this AI is still a confirmation or a binding interpretation.

It remained a confirmation and was approved by 10-0-2 with the above changes.

AI-104 -- Version and Body_Version attributes

The discussion exposed two conceivable ways of determining when a new version of a software module has been created by the developer:
- when the software has been (re)compiled
- when statements in the software source code have been changed

In the first case, a time stamp of the (re)compilation time can be used as a version number. In the second case, checksums on source without comments and ignoring case and extra spaces can be used, which is what GNAT does. Norm recommends that identified mechanisms (such as the ones above) be moved/added in an enumerated list to the paragraph in Discussion section that begins "The mechanism for ensuring...".

Robert Eachus provided a very complex example that caused the group to consider the impact of a renaming declaration of a library unit on this. It would appear that Annex E says that the 'Body_Version of the new unit takes the 'Body_version of the renamed unit and therefore has no impact on this AI. However, the manual isn't really clear on this. Bob Duff will put some additional discussion in the AI to show that the AI recommendations work in the presence of renaming.

Robert Dewar suggests that units without completions should not have a 'Body_Version value different from all others (as stated in the Summary) but the version value of the corresponding spec.

And then there were presentation comments pertaining to the size of the Summary section and the lack of the Wording section. Bob will add "see Summary" in the Wording section.

The discussion moved to understand the use of these version values to the programmer. In particular, none of the attendees could come up with an example in which the 'Version attribute could yield useful information, given the consistency checking required by Section 10. Input will be sought from the Distribution Annex authors and implementers to provide guidance in this area.

Provisionally approved (12-0-0) subject to editorial review, and a specific review and contributions by Annex E authors.

On the next day, Robert Dewar returned to this AI, showing an example where changes in subunits might be construed to cause a change in the value of 'Body_version of the enclosing bodies.

The AI is then tabled, confirming that we really need input from authors and implementers of the Distributed Systems Annex to provide guidance in this area.

AI-106 -- Freezing Rules

Robert Eachus is alarmed that the freezing rules might imply a runtime check.
His example of the problem is

```
package A is
  type T is tagged record
    I: Integer; ....
  end record;
  type A is access T'Class;
  function F(...) return A;
end A;

package B is
  type C is new T with private;
  AV: A.A := F(3);  -- (1)
  AI: Integer := AV.I; -- (2)
...
end;
```

Is C frozen at (1) or (2)? No. There is neither a rule nor a reason for freezing C, since code for (1) and (2) can be generated without freezing C. In fact no value of C is involved in the evaluation of F or the dereference of AV. The conclusion is that his example is not a problem.

The AI is approved (11-0-0).

AI-107 -- Base attribute for non-scalar subtypes?

The change requested by Norm Cohen had been reflected in the AI. Norm is happy.
Approved by 12-0-0.

AI-109 -- Size and Alignment Attributes for Subtypes

This AI was tabled jointly with AI-51.

AI-110 -- No Constraint Check on 'out' Parameter of an Access Type

The editorial change was limited to removing the superfluous ">" characters in the quotation of the AARM.reference. Approved 12-0-0.

AI-112 -- Wide_String file names?

Changes were aimed at depersonalizing the Question section. The question mark was removed in the new title. Approved 12-0-0.

AI-117 -- Calling Conventions

There was considerable discussion on the special rule defining implicitly declared dispatching "/=" operator to be intrinsic. This led to a hopscotch journey through the reference manual to (successfully) find conclusive evidence to show that this part of the AI is a ramification.

The binding interpretation portion deals with the inheritance of calling conventions for inherited or overriding subprograms. Robert Dewar questioned the motivation for this revised rule (as opposed to the need for explicit Convention pragmas, where needed, under current rules). This was countered by the observation that such pragmas would in fact be required for all the inherited/redefined subprograms, while the pragma argument was limited to the convention of the original subprogram anyhow, i.e., this would be strictly busywork for the programmer. The argument that this might be good documentation foundered under the observation that usually an entire type with all its subprograms would share the same convention and umpty convention pragmas would not really be so beneficial after all.

Bob will rewrite the AI with these points in mind and a ballot will be taken on the AI.

AI-118 -- Termination signals query of Terminate attribute

The issue is the validity of global variable values that are manipulated by task T1 and read by a task querying the termination of T1. We discussed whether T1'Terminated should be a synchronization point between the querying task and T1. This was quickly resolved to be an inappropriate model, since the query is asynchronous to T1 and hence the notion makes no sense in this generality. The signaling notion, on the other hand, seems the right model, but do we want to live with the cost implied ? We then returned to the intent of the AI. Should it be the case that, if the attribute yields true, the querying task can be assured that global shared variables have reliably updated values ? This implies not only that, upon termination, a task updates its cached globals (no disagreement here), but also that the querying task needs to refresh its caches. If T1 is not terminated, then accesses by T1 are in any case subject to the usual hazards of unsynchronized access to shared data.
The focus of the discussion was whether the efficiency cost of requiring such cache refreshes was warranted, given that most uses of the 'Terminated attribute can be expected to not be (ab?)used with this intent. A secondary question was whether the query will always refresh registers and caches regardless of its returned value or will do so only for a "true" as result. It was argued that even the "false"-case of conditional refreshing impacts code quality, since register management will have to assume the worst of both cases after the query. Robert Dewar notes that this worry is moot for GNAT in light of the flushing that is always done for runtime calls, which T'Terminated can be expected to be.
It was noted that this is another piece of evidence that programming with shared variables is VERY ADULT programming and should be carefully considered before being used.

After lengthy discussion, the current version of the AI was approved (9-0-3) with the editorial correction of adding an "end" to the example.

AI-121 -- Pragma Attach_Handler on Nested Objects

The issue is the presentation of the Summary, where the description of the problem should be in terms of the semantics of program execution as opposed to placing requirements on the programmer. Erhard suggested the following new wording for the first sentence, which was approved:
A program execution is erroneous if the handlers for a given interrupt attached via pragma Attach_Handler are not attached and detached in a stack-like (LIFO) order.

The second sentence remains unchanged.

Approved 11-0-1.

AI-123 -- Equality for Composite Types

The discussion focused on paragraph 2 of the Summary, namely whether equality of System.Address composes. At the previous meeting, there was some reluctance to include Address in the list of types for which equality composes. The discussion determined that the problems created for the programmer by excluding this type far outweighed the hypothetical consideration of bizarre address representations. On almost all systems, address equality will be defined as a bitwise comparison. Erhard proposes that System.Address be added to the list of types that compose correctly. Norm recommends that now the Summary should state that the correct composability applies to all language-defined types with equality operator. Approved 10-0-2 with these changes.

AI-124 -- Ligatures Are Allowed in Identifiers

Pascal objects to the Summary because it uses "ligature" which is an outdated term by a Corrigendum to ISO 10646. He recommends that the Summary be stated so that the capital and small AE are letters and therefore be allowed in identifiers. It remains a binding interpretation because it forces implementations to bind to the corrected ISO document, while the RM cites the original 10646. Bob rewrote this AI for a final vote on the next day when it was approved 11-0-0.

AI-126 -- Remote_Types Packages

Bob did analyse all the predefined packages in preparing this revision. He was concerned that Ada.Exceptions is not on the list. Furthermore he believes that it is important that Ada.Finalization is included in the list for safely cleaning up remote types.

Robert Dewar proposed that any additional classification of packages be implementation-defined to expedite the release of this AI because the complete analysis is time-consuming. Furthermore, implementations will do the "right thing" for their implementations which may conflict with a list that strives for completeness. The opposing voices mentioned the obvious portability consequences of such implementation-definedness.

The Ada.Finalization package is categorized as a Remote_Types package. Robert Eachus is concerned that there is a hole in the model of Controlled types that makes this a potential problem, while others contend that careful analysis by Pazy and Gargaro and experience of GNAT distributed systems implementers has not uncovered a hole like this.

Robert's nervousness over the other packages was alleviated with the promise that the AI should get close analysis from implementers.

Robert Dewar recommends that the GNAT Distribution team be invited to the London meeting to offer comments on the distribution-specific AIs.

The specific changes to the AI were enumerated as follows:

- The title should be changed to reflect the more global scope of the AI
- Elevate the cases of additional Pure pragmas to the summary section
- Remove the "???"-paragraph from the exception paragraph in the !discussion section
- Remove the "I guess" comments in the list of packages

The Remote_Types classification of Ada.Finalization should be under particular scrutiny by the implementers and any such decision should depend on the feedback received. Robert Dewar will "poke" his team to take a look. Other implementers are strongly encouraged to do likewise.

Approved 8-0-2. A letter ballot (and input from implementers) is requested.

AI-127 -- Expected type of a 'Access attribute

Latest changes handle the comment from Norm during the ballot and are confirmed by the meeting. Approved 8-0-4.

AI-128 -- String Packages

An objection had been raised about point 2 of the Summary, whether an upper bound that is greater than the actual upper bound yields an exception and whether Constraint_Error is raised instead of Index_Error. This has real impact on the ACVC tests. It seems that some existing compilers (such as, Thomson and GNAT) already handle the situation as described by point 2. If point 2 is accepted, then this AI becomes a Binding Interpretation because the description of the Slice subprogram is silent on this point. Gary proposes that the AI include a reference to Annex A. Also, point 4 of the summary should end with "paragraphs of the RM".

Approved 12-0-0 as a Binding Interpretation with these editorial changes.

AI-131 -- Interface to C -- passing records as parameters of mode 'in'

The AI was viewed by some as going overboard; the change should apply at best to types with pragma Convention(C) applied. Tuck also commented that it must not apply to tagged (record) types. In favor of the AI (duly modified) it was argued that the parameter convention

of C is indeed "by value" for structs and that "by reference" could always be achieved on the Ada side using 'Access, while a "by-value" cannot be achieved by any means, if the AI is not passed. Against the AI it was argued that, although the current Implementation Advice is bad advice, it is being followed by implementations. The situation is not "sufficiently wrong", the AI shouldn't tamper with this situation. Besides, few C programs take structs as parameters; they usually take pointers to structs. There is opposition to this latter statement, citing occurrences of structs in signatures of standard bindings. Tuck has recommended a way of passing records and blocks of data by copy, using two new pragmas, C_pass_by_copy and Convention (C_pass_by_copy, record_type); implementations could pick up these pragmas to give users "by-value" passing conventions.

Several alternatives are thus being considered:

1. Do nothing and turn the AI into a confirmation of the RM, elimination most of the Discussion.
2. Restrict the current AI to types covered by C Convention.
3. Confirm the RM with additional wording recommending a pass-by-copy pragma.

A straw vote of 7-0-5 gives direction to Bob to write up the third alternative.

AI-132 -- Exception Raised at End of Stream

Bob has rewritten the AI to present four alternatives for handling the end of stream. We need to pick one alternative for the Summary.

Several people argue that it is important to check with implementations on how they handle this particular situation, such as OCS, GNAT, Intermetrics, so that the approved alternative doesn't choose semantics that run counter to the majority of implementations.

As in the previous meeting, much of the discussion focused on the meaning of "end of input" for a stream and on whether it was necessary, important or even desirable to distinguish a situation, in which no more data can be obtained from a stream, versus one, in which insufficiently long input can be obtained (e.g., to fill some, but not all components of a record or array).

There was some discussion verifying uses for streams in Ada:
- Stream_IO
- Partition Communication Subsystem
- Abstract communication gizmo, such as support for sockets.

An apparent conflict has been uncovered on this issue. Paragraph 13.13.1 (8) states that a T'Read will produce no exception in the presence of an end-of-file but that the number of components read (in Last) is less than the length of the Item array. Paragraph A.13(12) says that a read past the end-of-file produces an End_Error exception. This apparent contradiction in the RM has produced much of the confusion.

The discussion turned to this question: Does the user expect the end of stream when doing a read or does the user have the responsibility to test for end of file when stream is mapped to

stream_IO?. If the user has this responsibility, can stream conversion to Stream_IO be done easily to get to the end-of-file function for the I/O? Apparently they can as follows:

```
if not End_of_file (Infile) then
  T'Read (Stream (Infile), Value)
        -- infile is stream file and is the implementation of the stream
```

[Note: the example used at the meeting applied a conversion to Root_Stream_Type to Infile. This would not match the signature of T'Read and hence I changed the example. E.P.]

In the end, much of the problem is due to not knowing how the stream is implemented

- If the implementation of the stream is a real file, then end-of-file translates into end-of-stream and end-of-stream has meaning.
- If the implementation is a buffer, then end-of-stream is not meaningful.

Under these circumstances it appears that the best semantics for this issue is:

1. If the data read is invalid, then Data_Error is raised.
2. If there is no data or not enough data, then End_Error is raised (Note that, if the user wants to know more detail, he/she can drop down to the file level and do EOF tests as shown above.)

This meeting approved the intent of alternative 3 by 10-0-1 and Bob Duff will update the AI for further review.

AI-136 -- Placement of Program Unit Pragmas in Generic Packages

Two small wording changes have brought this back to our attention.
Approved 12-0-0.

AI-141 -- Exceptions in Interfaces.C and its Children

The AI deals with a small problem that exposes a larger organization/architecture problem with the positioning of exception declarations in standard libraries. Do we fix this problem in the architecture now, barely in time, or is it already too late, these packages being standard bindings to C code ? The last meeting encouraged with overwhelming majority an AI that would fix this architectural problem.

While there was consensus in acknowledging the technical justification of the revised AI, there was considerable discussion, disagreement and rhetoric on the political side of making such a change to the language. Two straw-votes taken during the discussion showed a growing majority against the language change proposed by the AI.

The second part of the Summary section was closely examined by Robert Eachus and seemed to withstand close examination.

Further discussion and a vote was tabled until the third day, when the AI was voted down as written by 2-5-3. The AI will "revert" to a previous version and merely answer the original question about the exception raised by Virtual_Length.

AI-145 -- Profile of predefined operators for scalar types

This AI resolves an inconsistency between sections 4.5.2 and A.1. There was considerable discussion about the difficulty of explaining the notion of Boolean'Base to the unsuspecting programmer but, in the end, consensus on the AI was unavoidable. The following changes were agreed upon:

1. add appropriate references to the header of the AI
2. change the classification of this AI to a binding interpretation
3. insert "...subtype-conformant {but has to be mode-conformant}" in the last line of the summary

Approved with these changes (7-0-3).

AI-147 -- Optimization of Controlled Types

There was reasonable consensus about the intent of the first part of the summary, i.e., that Initialize and Finalize calls on otherwise unused variables of limited controlled types are not to be eliminated.

There was also reasonable agreement on allowing the optimization of eliminating temporaries (and the Adjust/Finalize pairs) not just for assignment, but all other constructs involving the copying of controlled values (as shown in the Question).

For non-limited types, the issue of eliminating Initialize/Finalize pairs is much more controversial. Robert E. argued that in the simulation domain quite a bit of the application semantics might be captured in the Initialize and Finalize subprograms, so that their elimination might be a problem there. The counter argument to this view is, for Finalize, that the unknown number of various finalizations of temporaries during value copies is likely to interfere with this strategy most severely anyhow. Erhard poses the question whether high-level analyses of source code that establish Set-Use connection must take such implicit calls as Initialize, Adjust and Finalize into account (which is quite a burden, especially given the dispatching nature of such implicit calls). He also reiterates earlier considerations in C++ compiler construction, where elimination of unneeded implicit calls really make a difference in efficiency of the code.

Somebody suggests that an implementation-defined pragma might be the right solution to control the (non-)elimination of the Initialize/Finalize pairs and that such a pragma might be coupled with control over garbage collection, a capability direly needed.

A vote to allow the elimination of Initialize and Finalize calls for non-limited types results in a 5-1-4 agreement. This AI was then tabled, waiting for a completed write-up.

AI-155 -- Stream-Oriented attributes for language-defined private types

Superseded by AI-108. AI-155 is now deleted.

AI-157 -- Visibility of Inherited Private Components

Erhard points out that there are different rules for creating the visibility to inherited operations and to components of derived types. There needs to be an editorial change in the AI to reflect this difference. Bob Duff disputes this statement and points to 3.4(10-14), where the necessary implicit declarations are specified to occur.

It was requested that the example really should match the Summary in its choice of unit names. It was also observed that the current example may not ideally illustrate the issues due to distractions from generic and abstract types. So it was recommended that a better, simpler example be chosen. Also Stephen Michell's example should be added:

```
package P is
  type Pt is tagged private...;
private
  type Pt is tagged record
    C: Integer;
  end record;
end P;

with P;
package Q is
  type Qt is new Pt with record
    C: Integer;
  end record;
end Q;

with Q;
private package P.child is
  type LocalT is new Q:Qt with record ...
  X: LocalT;  -- X.C refers to the C component of Qt
                -- but X can be view-converted to get at the C component of Pt.
```

The intent of the AI is approved 12-0-0. Substantial editorial changes are necessary and the result will be subject to editorial review and a vote.

AI-158 -- T'Class as generic actual type

Tuck wants to add more meaning to the case of when T'Class is passed as actual type to a generic formal type, so that renaming of its primitive operations can be done with dispatching being performed when referring to the renamed operations. In particular, dispatching operations should match (implicitly) with generic formal subprograms.

It appears that Tuck's proposal doesn't break anything and provides an opportunity for a apparently useful extension, as opposed to closing a hole (the semantics of renaming primitive operations in a generic) with an illegality interpretation.

Input from implementers may be useful to determine if implementations could be unduly affected by the general notion proposed by Tuck. There was a question whether this AI could impact the ACVC and we concluded that the answer was no.

Gary Dismukes stated that the user could accomplish the same goals with wrappers, albeit with much more work. Most members at the ARG see lots of value in the proposed capability, yet now the question is to determine whether there is sufficient need that justifies this extension. It was also noted that an AI phrased as a general overhaul of the dispatching model has very little chance to get by the ARG, let alone WG-9.

Bob will complete the AI with (or without) the assistance of Tuck. Any further consideration or votes await the completion of the AI.

AI-163 -- User-defined fixed-fixed multiplying op

It was noted that this incompatibility was known during the design of Ada95 and was deemed acceptable in exchange for the user convenience of not having to apply explicit type conversions to every result of a fixed-point operation.

Consequently the ARG decided (10-0-1) that this AI be written as a confirmation of the existing rules.