Minutes of Electronic ARG Meeting 62H

9 December 2020

Attendees: Raphael Amiard (after 10:40), Steve Baird, John Barnes, Randy Brukardt, Arnaud Charlet, Jeff Cousins, Gary Dismukes, Claire Dross (until 12:15), Bob Duff, Brad Moore, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

Observers: None.

Meeting Summary

The meeting convened on Wednesday, 9 December 2020 at 10:32 hours EST and adjourned at 13:26 hours EST. The meeting was held using Google Meet. The meeting covered the majority of the agenda.

AI Summary

The following AIs were approved with editorial changes:

AI12-0330-1/04 Add items to the glossary (14-0-1)
AI12-0401-1/03 Remaining of a qualified expression of a variable (15-0-0)
AI12-0402-1/02 Master of a function call with elementary result type (12-0-2)
AI12-0404-1/05 Presentation issues from Draft 26 review (15-0-0)
AI12-0405-1/01 Fixups for stable properties (8-0-6)
AI12-0406-1/02 Clarifying static accessibility (15-0-0)
AI12-0407-1/03 Fixups from Draft 26 review - part 1 (15-0-0)
AI12-0408-1/02 Definition of “null procedure” and “expression function” (15-0-0)
AI12-0409-1/01 Preelaborable_Initialization and Bounded Containers (11-0-3)
AI12-0413-1/01 Reemergence of "=" when defined to be abstract (15-0-0)
AI12-0414-1/00 Replace categorization pragmas with aspects (15-0-0)

The intention of the following AI was approved but it requires a rewrite:

AI12-0412-1/01 Pre'Class using an abstract function applied to a concrete operation of abstract type (13-0-2)

Detailed Minutes

Welcome

Steve welcomes everyone. He describes us “As the Language Turns”, which gets a general laugh.

Apologies

Jean-Pierre Rosen sent apologies; he had a class to teach during the meeting period.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 14-0-0.

Date and Venue of the Next Meeting

Following our approximately 6 week schedule, Randy had suggested Wednesday, January 20th for our next meeting. That is after the WG 9 meeting, our goal is to have something before then. Randy notes that we can continue to make fixes until ITTF publishes the Standard, and more practically until WG 9 votes to send it to SC 22. WG 9 hasn’t even created a project yet, and also will need some internal review of what we send them. Also, as a practical matter,

Randy notes that he still has a number of reviews to process. Tucker suggests that we deal with those with e-mail votes.

We stick with Wednesday, January 20th for our next meeting, same time and platform.

Unfinished Action Items

There are two unfinished action items (Steve Baird, AI12-0016-1; Tucker Taft: OpenMP Technical Report). Tucker notes that he gave a talk on OpenMP at HILT.

Current Action Items

The combined unfinished old action items and new action items from the meeting are shown below.

Steve Baird:

- AI12-0016-1

Randy Brukardt:

Editorial changes only:

- AI12-0330-1
- AI12-0401-1
- AI12-0402-1
- AI12-0404-1
- AI12-0405-1
- AI12-0406-1
- AI12-0407-1
- AI12-0408-1
- AI12-0409-1
- AI12-0413-1
- AI12-0414-1

Tucker Taft:

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.
- AI12-0412-1
- Glossary entry for “Accessibility”, see discussion of AI12-0406-1.
- AARM Note for AI12-0402-1 to explain reasons for the composite-only rule.

Detailed Review

The minutes cover detailed review of Ada 2012 AIs (AI12s). The AI12s are presented in numeric order, which is not necessarily the order in which they were discussed. Votes are recorded as “for”-“against”-“abstentions”. For instance, a vote of 6-1-2 would have had six votes for, one vote against, and two abstentions.

If a paragraph number is identified as coming from the working Ada 202x AARM, the number refers to the text in draft 27 of the Ada 202x AARM. Paragraph numbers in other drafts may vary. Other paragraph numbers come from the final consolidated Ada 2012 AARM; again the paragraph numbers in the many drafts may vary.

Detailed Review of Ada 2012 AIs

AI12-0330-1/04 Add items to the glossary

Steve would like a minor change in the glossary entry for Suppress. He suggests:

> To suppress a check is to give a request to the implementation {(or at least permission)} to disable a run-time check for a portion of the text of a program.

The parenthetical remark is weird. Tucker suggests just replacing the word "request", giving:

> Suppress
> To suppress a check is to give a [request]{permission} to the implementation to disable a run-time check for a portion of the text of a program.

Bob objects, it should be a request in most cases (excepting cases where hardware is doing the check for free). Richard thinks it is a permission. Tucker says that "request" suggests that the compiler is screwing up if it doesn't remove the check, but that's not the case. You don't want anyone to assume the check will not happen.

Bob offers to send a suggestion.

Ed notes a spelling error: A "logicial"{logical} thread of control.

Tucker's accessibility entry will get added here (see discussion of AI12-0406-1; that AI was discussed first).

Approve AI with changes: 14-0-1. Arnaud abstains.

AI12-0401-1/03 Remaining of a qualified expression of a variable

Randy is asked why this one is back. There were four significant changes, detailed at the end of the !appendix; it seemed like too much change to do without a revisit.

Ed notes "incompatable" in the discussion.

Gary comments that in the summary "even then" should be "even though".

Approve AI with changes: 15-0-0.

AI12-0402-1/02 Master of a function call with elementary result type

This is simplified to just except returns of elementary types.

Steve asks about a corner case. In the case of an access-to-function type, is it clear what point is the "point of function declaration"?

Gary would like an AARM note to explain why this applies only to composite types. Tucker volunteers to write that.

We add some wording for Steve's question (and a "the" for Gary):

> ...of {the} function {(or access-to-function type)} declaration…

Steve muses whether this applies to generic formal function. A generic formal private type is formally composite, but the actual may not be composite. Tucker says that's fine, it is the safe direction. (It's OK to apply legality checks to a call to a body that doesn't expect them to be made; it's not OK to omit legality checks for a call to a body that does expect checks to be made.) A similar issue occurs for private types completed with elementary types.

Approve AI with changes: 12-0-2. Ed and Jeff abstain.

AI12-0404-1/05 Presentation issues from Draft 26 review

Richard Wai: in wording 5.6.1, there is an extra “chevron” in the text. This leads to a long and rather pointless discussion about whether that is really a chevron, one that is sideways, or whether that is an incorrect characterization.

Jeff: There doesn’t seem to be any wording for (10). [Editor’s note: The !corrigendum sections exist, but as usual with the original paragraph numbers, which are very different in this case. There was no wording.]

Jeff: D.3(13): :('now' is obtained => ‘now’ being obtained or ‘now’ having been obtained. Tucker and Jeff think this is OK as is.

Approve AI with changes: 15-0-0.

AI12-0405-1/01 Fixups for stable properties

Steve goes over the questions.

Arnaud suggests that this feature is not ready, noting all of the issues. Several people agree in principle. Others think that this is a useful feature for which we’ve already done most of the work. We move on without deciding this point.

For (A), it probably isn’t worth it to require a stable property to be given, especially given that we want to stay simpler.

For (B), Tucker argues that “anonymous access” are special pass-by-reference regular parameters in usage (he gives the example of the special semantics in prefix views); that is rather different than named access types. They also are controlling parameters. Others feel that named access and anonymous access should be the same. Tucker drops this one in order to move on.

For (C), use the equals operation as defined for memberships. Randy drops his objection as this is rather unlikely to happen (only for scalars/arrays with redefined “=”).

For (F), Randy notes that we wanted people to explicitly say *something* about stable properties. Either a **not** property definition, or something in the postcondition. Some unrelated mention in the postcondition is not enough for that. This was discussed at an ARG meeting when the original wording was reviewed. Tucker agrees with Randy. So stick with the status quo.

Steve notes that there are some empty headers in the !appendix, those should be removed.

So in the wording, delete proposal A.2, B, F.2.

Steve notes that this is a fix-up AI, we’ve already approved including the feature in Ada 202x.

Approve AI with changes: 8-0-6. Abstain, Gary, Arnaud, Justin, Jeff, Brad, Ed.

AI12-0406-1/02 Clarifying static accessibility

Steve wants the order in the glossary entry changed:

> When a master is left, associated tasks are awaited and associated objects are finalized.

!example: final paragraph. “A dynamic check is still require{d}[s] at {(1)}[one], …”

Glossary for “master construct”: A master construct is one of certain executable constructs {listed in 7.6.1} ...

Ed would like something to be mentioned about accessibility in these. Randy objects, noting that masters are interesting without involving accessibility; it is the definition of accessibility that depends on masters, not the other way around.

Ed then suggests that we have glossary entry for “accessibility” that involves masters. Randy checks and notes that there isn’t one. It does seem like something that should be there. Tucker volunteers to write one. This will be added to AI12-0330-1 (discussed after this AI).

Brad: 7.6.1(3/2), second sentence: should the “:” be a “;”. No, the colon introduces the list separated by semicolons.

Randy notes that we need to discuss the second issue a bit. He notes that he switched the wording to bullets rather than duplicate the lengthy sentence about “not considered to be statically deeper, nor statically shallower”. This meets general approval.

Approve AI with changes: 15-0-0.

AI12-0407-1/03 Fixups from Draft 26 review - part 1

(3) Summary: This should not say “representation aspect”, as it remains an operational aspect. Randy explains that the purpose of this aspect requires it to be specifiable for private types, but representation aspects are not allowed on those.

Are any of these controversial? Randy goes through the list and explains each. Only the freezing one (#6) could be. The problem was that there were cases not covered. Does anyone want to discuss freezing rules in detail? A long silence ensues, and there is no discussion.

Jeff: 12.3(15) there is unbalanced parens. “… 12.6, "Formal Subprograms"{)}. ...”

In the !problem, (5), second line: There is {a} To Be Honest…

Ed: In (6) “adding additional” is redundant, drop “additional”. Someone says that there are seven AIs with the phrase “adding additional”. We aren’t going to try to fix the other six. [Editor’s note: This isn’t really redundant, it is just a shorthand for “adding additional Pre and Post assertions to those of the actual”. I added (which are then added to those of the actual) while deleting “additional”, since it is the reason for this item in the AI.]

Approve AI with changes: 15-0-0.

AI12-0408-1/02 Definition of “null procedure” and “expression function”

Tucker explains the AI.

Steve suggests adding “on” to 6.7(4/2):

> The execution of a null procedure is invoked by a subprogram call. For the execution of a subprogram call on a null procedure{, or on a procedure completed with a null_procedure_declaration}, the execution of the subprogram_body has no effect.

We need to do the same in 6.8(7/4).

Also in 6.8(7/4) there is an extra word:

> The execution of an expression function is invoked by a subprogram call. For the execution of a subprogram call on an expression function{, or on a function completed with a expression_function_declaration}, the execution of the subprogram_body executes an implicit function body containing only a simple_return_statement whose expression is the return expression [that] of the expression function.

The !subject should not end with a period.

!received is missing the day, it should be 20-10-27.

Approve AI with changes: 15-0-0.

AI12-0409-1/01 Preelaborable_Initialization and Bounded Containers

Tucker explains the problem and the fix.

Steve wonders why we only talk about types in a package specification. That was a requirement of the original Preelaborable_Initialization pragma.

Ed wonders if this should apply to the bounded queues. Those have entries so they don't have (and cannot have) Preelaborable_Initialization.

Bounded_Indefinite_Holders should also have this, A.18.32.

Randy suggests that we don't allow attribute_definition_clauses for this aspect. It's work and one would never want to use one, since these aspects have to be visible in order to be useful. He thinks attribute_definition_clauses are only useful in new code to hide representation aspects in the private part, and are otherwise obsolescent. Others think that it needs to be consistent, otherwise there would be a situation where an attribute could not be specified with an attribute_definition_clause. [Editor's note: Isn't that true of most attributes, such as First, Last, Succ, and Image?]

Straw poll: Should we allow specification by attribute_definition_clause: In favor: Richard, Steve, Justin, Tucker, Ed, Brad, Gary, Bob (8). Opposed: Randy (1). Abstain: everyone else (5).

Typo: "argubly" in the !problem.

Randy notes that we need to answer this question. Should this be a Binding Interpretation? We should allow Ada 2012 implementations to support this, but not require this. So make it a Binding Interpretation, but don't require P_I aspects in Ada 2012 ACATS tests.

Approve AI with changes: 11-0-3. Bob, Arnaud, Gary abstain.

AI12-0412-1/01 Pre'Class using an abstract function applied to a concrete operation of abstract type

[Editor's note: The subject of this AI is too long; the author should try to shorten it.]

Steve asked about Do_Stuff'Access, and about Do_Stuff passed as a generic formal parameter. Tucker suggests that these would need to raise Program_Error.

One would not want to call these things directly anyway. It's dubious to call routines with objects of an abstract type. Randy says we could make all such calls illegal. Tucker says that would have too much risk of breaking something for someone.

Steve wonders if renaming gets affected.

Claire notes that one cannot give a default for a function for an interface. If you could, she thinks this problem wouldn't exist. Tucker notes that interfaces can be combined. And it is useful to require something to be overridden. If you require a default to exist to use Pre'Class, then you are forced to choose between having Pre'Class or requiring overriding. We try not to require users to make such choices.

"Extensions_Visible" is a GNAT thingy that isn't necessary so it should be removed from the !question.

We'll try to determine if we can make 'Access illegal. Tucker will take this back.

Approve intent: 13-0-2 Ed, Jeff abstain.

AI12-0413-1/01 Reemergence of "=" when defined to be abstract

The example in !question is missing some new-lines around "package Gen is".

Tucker explains the rules.

Randy wonders why the 12.5 rule is only about “untagged records”. Tucker says that tagged types use a different mechanism for definition, so the predefined and primitive operators are generally the same (the predefined operators call the correct primitive operation).

In the question: (Neither; {because}[if] T is an …

Brad was wondering about the situation for arrays. He would like to add that to the question. Randy objects that questions shouldn’t be embellished with things not asked, and Jean-Pierre only asked about scalar types.

Randy suggests adding a paragraph to !discussion to talk about array types and record types, and remove the record part of the !question.

Approve AI with changes: 15-0-0.

AI12-0414-1/00 Replace categorization pragmas with aspects

Randy explains that this AI is mainly a holder for the list of paragraphs that will be changed.

Gary notes that the Summary needs a period at the end.

Randy will add the paragraph numbers associated with the changes when he updates the Standard Draft.

Approve AI with changes: 15-0-0.