Minutes of Electronic ARG Meeting 62J

20 January 2021

Attendees: Raphael Amiard, Steve Baird, John Barnes (had trouble connecting so after 11:07), Randy Brukardt, Arnaud Charlet, Jeff Cousins, Gary Dismukes, Claire Dross (until 11:53), Bob Duff, Brad Moore, Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

Observers: None.

Meeting Summary

The meeting convened on Wednesday, 20 January 2021 at 10:32 hours EST and adjourned at 12:55 hours EST. The meeting was held using Google Meet. The meeting covered the entire agenda. We took a 20 minute break at 11:53 in order to watch the inauguration of the new US President.

[Editor's note: Per our usual policy, we do not use 'I' or 'O' with numbers, so there is no meeting 62I.]

AI Summary

The following AIs were approved with editorial changes:

AI12-0411-1/01 Add "bool" to Interfaces.C (15-0-1)
AI12-0412-1/05 Abstract Pre/Post'Class on primitive of abstract type (14-0-1)
AI12-0416-1/05 Fixups from Draft 26 review – part 2 (16-0-0)
AI12-0417-1/01 Make categorization pragmas obsolescent (16-0-0)
AI12-0418-1/04 Presentation issues from Draft 26 review – part 3 (15-0-0)
AI12-0419-1/01 Aspect inheritance and reemergence (16-0-0)
AI12-0422-1/02 When is a constant known-on-entry? (16-0-0)
AI12-0423-1/01 Aspect inheritance fixups (15-0-0)
AI12-0424-1/01 Concurrency and the standard storage pool (10-0-5)

The following AIs were discussed and placed on hold:

AI12-0410-1/01 Storage Pool-related side effects (11-1-3)
AI12-0415-1/01 Parallel_Calls aspect for types (15-0-0)
AI12-0420-1/01 Preelaborable_Initialization and contracts (15-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

No one sent apologies.

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, March 3rd for our next meeting. We briefly discussed this and decided to wait on this until the end of the meeting.

At the end of the meeting, we decide not to schedule another meeting at this time. We need to wait for WG 9 review comments.

ARG Approval of Draft

Arnaud says that he thinks that we need no additional review of the RM draft. He thinks that trusting the editor makes sense. WG 9 is planning a review period during which ARG members can provide additional input if needed, and that includes making sure the editor didn't make any gross errors.

We decide to wait until the end of the meeting to address this topic.

At the end of the meeting, we come back to this topic. Most suggest that we should approve the draft to send to WG 9. Randy is directed to request specific reviews of particular questions from other ARG members if there are issues (most likely integration issues) that he needs help with.

New issues, including language bugs not related to existing AIs should be deferred until after Ada 202x.

A deadline is requested. Randy is asked how long he thinks it will take to finish the draft. He says at least two weeks. A deadline of the end of February is suggested (this gives him extra time).

Vote: Send draft to WG 9 when complete, no later than the end of February, with no further ARG review: 15-0-0.

We give ourselves a round of applause for completing Ada 202x. [It doesn't feel done for your editor, he's got several weeks of work to go. - Editor.]

Unfinished Action Items

There are two unfinished action items (Steve Baird, AI12-0016-1; Tucker Taft: OpenMP Technical Report). We did not discuss these.

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:

- Create wording for AI12-0417-1.
- Split the 13.11(17.1/5) change into a separate AI (assigned AI12-0424-1 after the meeting).

Editorial changes only:

- AI12-0411-1
- AI12-0412-1
- AI12-0416-1
- AI12-0417-1
- AI12-0418-1
- AI12-0419-1
- AI12-0422-1
- AI12-0423-1

Tucker Taft:

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.

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 28 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

AI12-0410-1/01 Storage Pool-related side effects

Tucker notes that we need to decide whom the side-effects of an allocator are "charged to" for the purposes of the Global aspect.

Arnaud thinks this is too complicated for a corner case. He thinks there will be more corner cases to be worked out. So he says it is no for him. Raphael agrees with Arnaud.

Steve notes that there is value in stating that there the standard pool is concurrent. He notes that this has been assumed since Ada 83, but it doesn't follow from the requirements of the language since the pool was made explicit in Ada 95.

Randy comments that we need to adjust the contract of Unchecked_Deallocation be something; it currently is the old Global. Tucker says it would have to be **in out all** if this is required.

Approve AI with changes: 5-3-7 Opposed: Arnaud, Raphael, John (hates access types), Abstain: Gary, Justin, Jean-Pierre, Brad, Jeff, Ed, Bob. In favor: Steve, Tucker, Randy, Richard, Tullio.

It is suggested to split the standard storage pool rule into a separate AI. [This was assigned AI12-0424-1 after the meeting – Editor.]

Hold the remainder of this AI: 11-1-3. Opposed: Arnaud. He says that he thinks the AI should be completely killed. Abstain: Bob, Raphael, Ed.

AI12-0411-1/01 Add "bool" to Interfaces.C

Tucker explains the AI. A question is raised about compatibility. It has the normal use clause incompatibility (if someone has defined their own "bool" and has use clauses for Interfaces.C as well as their own packages), but otherwise people can avoid using it.

Brad, in the second paragraph of discussion, there should be period at the end of "anything other than zero{. In}[, in] both". [Editor's note: For this to make sense, we also need to remove the "Although" at the front, and add a "However" before the "In". Otherwise, the first part is a sentence fragment.]

Arnaud mentions that he wants to make sure we stop adding new features. This one is simple enough but we need to finish someday.

Approve AI with changes: 15-0-1 John abstains.

AI12-0412-1/05 Abstract Pre/Post'Class on primitive of abstract type

Tucker explains the AI. He re-explains the notional formal derived type model – it makes the expression be meaningful for all descendants of the type.

In !discussion, “one original proposed” should be “one original{ly} proposed”.

Arnaud is concerned that it is adding complexity. Tucker comments that the existing wording has bugs and also makes things illegal that users want to do.

Ed wonders if there is an ACATS test on the old rule. He agrees that it should make the implementation simpler and more sensible for the user.

Claire would like additional rationale in the AARM. Tucker agrees, and will write this.

Later, Tucker read to the group what he wrote:

> AARM Discussion: The above rules mean that a concrete primitive of an abstract type is effectively treated as abstract, if any nontrivial Pre’Class or Post’Class aspects apply to it. This makes sense because we are using a notional formal derived type model for such aspects, and an abstract type is not permitted as an actual type for such a formal type.

Remove “extensions_visible” from the example in the !question.

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

AI12-0415-1/01 Parallel_Calls aspect for types

This is too ambitious at this point.

[Jeff sent a note through chat that in the examples Formal_Acc’(X) should be Integer’(X).]

Hold: 15-0-0.

AI12-0416-1/05 Fixups from Draft 26 review – part 2

Randy explains the rules that were changed and why.

Jeff notes two typos: First, in the wording, item 1, the last line of the AARM note: “Containers”. In the first line of ! discussion spelling: “whatever is convinient in”.

Approve AI with changes: 16-0-0.

AI12-0417-1/01 Make categorization pragmas obsolescent

Randy asks whether we want to do this.

John says that we should do it to be consistent.

Steve asks whether we can approve this without wording. He thinks we should trust Randy since this is rather mechanical. Bob agrees.

Tucker suggests having Steve and John review the wording when finished.

Brad notes that there are errors in the summary (“Obsolescence” should be “Obsolesce”) and the problem (“obsolescenced” should be “obsolesced”).

Approve AI with changes: 16-0-0.

AI12-0418-1/04 Presentation issues from Draft 26 review – part 3

Jeff: 6.1.1(8/3) is missing the {} to mark an insertion; it should be around the parenthesized text.

12.7(4.5/3) is missing the first letter: 'I'.

Brad noted that in the editor's question: "Argubly" should be "Arguably", and "chose" should be "choose". Randy notes that we should simply answer the question and delete it. Tucker says it isn't worth doing more than was already done. So delete the Editor's question.

Brad: 4.3.1(17.2/5) is paragraph 4.3.1(17.3/5) in draft 28 RM. Is it correct in the AI (which is based on draft 27)? Dunno, the editor will check. [It needs to be changed. - Editor]

Approve AI with changes: 15-0-0.

AI12-0419-1/01 Aspect inheritance and reemergence

Tucker notes that this is primarily presentation, but there is one question.

In the past, how aspects associated with subprograms are inherited seemed to be defined on a case-by-case basis. This means that there isn't much consistency in those rules, which makes everything more difficult for both users and implementers. So he set out to try to categorize the methods into just a few mechanisms.

He ended up with three mechanisms. He describes the three different ways that aspects can be inherited.

Steve notes that having a taxonomy makes it easier to understand.

The question is about user-defined literals. Currently, they're like stream attributes. Tucker believes that it is better in the nonoverridable category.

Jeff says that he agrees with Tucker, he thought that literals worked this way. We will choose Option 1, and Option 2 will be deleted from the AI.

Tucker did not try to make a general model of composition, as the exact meaning seems to depend on the aspect.

Brad notes a typo: "... untagged derived type, the ancestor's nonoverridable aspects, if any{,} reemerge, which is not surprising[,] because the primitive …"

Jeff says that there is a misspelling in 7.3.2(3/4): "Redudant"

Tucker: "overridding" in !discussion.

[Editor's note: "Nonoverridable" is defined in 13.1.1, so the cross-reference in 4.2.1(6/5) should be to 13.1.1, not 13.1.]

Approve AI with changes: 16-0-0.

AI12-0420-1/01 Preelaborable_Initialization and contracts

Randy notes that we don't have a solution to propose for this problem. Tucker says that we've lived with it (not actually noticing it) for Ada 2012, so we don't need an immediate solution.

Gary notes a misspelling: "Argubly".

Hold: 15-0-0.

AI12-0421-1/01 Legality checks for class-wide Pre and Post

Randy explains that after the agenda was published, he and Tucker realized that this topic was closely related to that of AI12-0412-1. So it was folded into that AI and then marked deleted. Thus, there is nothing to discuss.

AI12-0422-1/02 When is a constant known-on-entry?

The rule was breaking privacy; Tucker notes that every time we use a dynamic rule in a Legality Check, we're usually sorry.

John does not like that a form of "assume" is used twice in the second AARM note. Change it: "as all private types and private extensions {might}[are assumed to] have a controlled component."

6.1.2(10/5) needs a comma in existing text: "(which, for the purposes of this clause{,} includes all constants …"

Delete the "temporary discussion".

Gary wonders if "may" is OK, Tucker says that it is "may not" that we cannot use.

Approve AI with changes: 16-0-0.

AI12-0423-1/01 Aspect inheritance fixups

Randy explains that these two issues would have been put into the basic fixup AI had they been ready in time. Neither change should be controversial: the first is a wording clarification, and the second is just making common sense explicit.

Nonreturning should be No_Return in the example of !question.

Jean-Pierre says that that "that in an" should be "that is an" in the summary.

Approve AI with changes: 15-0-0.

AI12-0424-1/01 Concurrency and the standard storage pool

This was split from AI12-0410-1 (see that discussion). [The number was assigned after the meeting – Editor.]

Bob thinks this is redundant. We don't say specifically that, for example, object declarations can be executed concurrently – that's just part of the assumed requirements.

Tucker notes that for subprograms, the requirement is that they work concurrently so long as none of the parameters overlap. But in this case, we have (possibly) a single object – the standard pool – being passed to a routine that is invoked concurrently. So this goes further than the blanket requirement and thus it is best to make this clear.

Approved AI with changes: 10-0-5. Abstain: Bob, Ed, Raphael, Arnaud, John.