Minutes of Electronic ARG Meeting 62G

21 October 2020

Attendees: Steve Baird, John Barnes (starting at 10:54), Randy Brukardt, Arnaud Charlet, Jeff Cousins (starting at 10:38), Gary Dismukes, Claire Dross, Bob Duff, Brad Moore (starting at 10:40), Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

Observers: None.

Meeting Summary

The meeting convened on Wednesday, 21 October 2020 at 10:32 hours EDT and adjourned at 12:52 hours EDT. The meeting was held using Google Meet. The meeting covered the entire agenda and finished early.

AI Summary

The following AIs were approved with editorial changes:

AI12-0384-2/01 Fixups for Put_Image and Text_Buffers (9-0-5)
AI12-0398-1/02 Most declaration should have aspect specifications (13-0-1)
AI12-0399-1/01 Aspect specification for Preelaborable_Initialization (13-1-0)
AI12-0400-1/02 Ambiguities associated with Vector Append and container aggregates (13-0-0)
AI12-0401-1/01 Renaming of qualified expression of variable (13-0-1)
AI12-0403-1/02 Presentation issues from Draft 26 review (14-0-0)

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

AI12-0402-1/01 Master of a function call with no special result type (7-1-6)

The following AI was discussed and assigned to an editor:

AI12-0405-1/00 Fixups for Stable Properties

The following AI was discussed and voted No Action:

AI12-0384-1/02 Fixups for Put_Image and Text_Buffers (14-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

Justin says that he won't be voting as he was unable to prepare due to his dog's death over the weekend.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 12-0-0.

Date and Venue of the Next Meeting

Following our approximately 6 week schedule, Randy had suggested Wednesday, December 2nd for our next meeting. That conflicts with some other meetings. We try a variety of other dates; Jean-Pierre has a tutorial on the 9th but that is the only conflict for that day. We settle on Wednesday, December 9th for our next meeting, same time and platform.

Unfinished Action Items

There are three unfinished action items (Steve Baird, AI12-0016-1; Tucker Taft: OpenMP Technical Report; Storage_Pools for formal access types). We did not spend any time talking about 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
- AI12-0405-1

Randy Brukardt:

Editorial changes only:

- AI12-0384-2
- AI12-0398-1
- AI12-0399-1
- AI12-0400-1
- AI12-0401-1
- AI12-0403-1

Tucker Taft:

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.
- Provide case statement example for AI12-0401-1.
- New AI to handle storage pools for formal access types (from private conversation).

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 26 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

AI12-0384-1/02 Fixups for Put_Image and Text_Buffers

This is an alternative for AI12-0384-2, which we approved instead.

No Action: 14-0-0.

AI12-0384-2/01 Fixups for Put_Image and Text_Buffers

Tucker explains the changes.

Brad comments that “Buffer_Overflowed” has a bad connotation, he suggests “Buffer_Truncated” instead. Ed says “At_Capacity”. Tucker suggests “Output_Truncated”. “Text_Truncated” is suggested. We agree on that.

Brad says that we need a comma after “is currently defined” in !problem, (1).

Also, (4) “There seems {to be} no particular advantage...”

In !proposal, (2), “the the”!

!discussion, 3rd real paragraph “you are willing” should be “one is willing”.

Gary notes that the first paragraph of !summary sounds awkward: “Add operations to associate with a Text_Buffer an adjustable level of indentation.” Put the “with a Text_Buffer” at the end of the sentence.

Approve AI with changes: 9-0-5. Jean-Pierre, Bob, Claire, Arnaud, Gary abstain.

AI12-0398-1/02 Most declaration should have aspect specifications

Randy goes over all of the choices he made.

Steve had noted that the syntax for enumeration_literal_specification is a comma separated list, so adding an aspect_specification would be ambiguous. Randy proposes to remove this, as whatever fix is applied would make this not a simple change.

So the change to enumeration_literal_specification is removed, the discussion is changed to say NO, and the reason given is that the syntax would be ambiguous.

Brad has typos. The subject should be “allow” rather than “have”.

Steve would prefer that the subject start with “More” rather than “Most”. There are more noes than yeses in the AI. [The subject is talking about all declarations, not just those mentioned in the AI – Editor.]

Gary suggests dropping the ? from choice parameters.

Brad in second paragraph of problem:

> However, a few kinds of declaration do not {allow aspect specifications}, and at least some of those have …

Tucker notes in the same paragraph:

> … provide {a}[an] preferred alternative.

In the chunk_specification paragraph, “unambigiously” should be spelled “unambiguously”.

Jeff: First sentence of discriminant_specifications:

> {The} {s}{S}imilar {constructs}[declarations] component_declarations and formal_parameters both allow aspect_specifications.

Jeff in iterated_component_association:

> This is specifically the loop parameter of {the} traditional form of the array ...

Jeff asks if SPARK has any need for these currently. Arnaud says no, but he’s in favor of opening the door to future uses. Claire concurs. Tucker notes that no one needs to do anything to their implementation, as this is an option. If one has no aspects that can be specified on (say) a discriminant_specification, one doesn’t even need to add the syntax to their parser, as no legal program could tell the difference.

Approve AI with changes: 13-0-1. John abstains.

AI12-0399-1/01 Aspect specification for Preelaborable_Initialization

This is a weird hole in the aspect model. Tucker found that it would require not requiring matching between the partial view and the full view.

Jeff notes “specified as True” and “specified True”, should be consistent. He says that there are 9 “specified True” and 5 “specified as True” in the current RM. Make all of these “specified True”.

Approve AI with changes: 13-1-0. Bob is opposed, he’s opposed to the cosmetic part of this. John says he’s already changed his book.

AI12-0400-1/02 Ambiguities associated with Vector Append and container aggregates

Tucker reminds us of the problem: if the element is a record or array type, then Append is now ambiguous as containers now have aggregates.

Brad: last paragraph of the summary and recommendation are essentially the same, can we get rid of one of them? Doesn’t seem to be a problem (so long as they say the same thing).

Gary: “Ada 202X” should be spelled “Ada 202x”. Add commas and a “the” in first paragraph of !recommendation:

> Eliminate the overloading of the vector x vector operations for Append, Insert, and Prepend, by adding the suffix "_Vector" instead of overloading. At the same time, we propose to eliminate the new Append_One routine{,} which was created to match the requirement of the Aggregate aspect for a two-parameter procedure to do appending. We eliminate the need for Append_One by changing the current Append{,} which has three parameters, {the}[that] last of which is defaulted, to be two separate overloadings, one taking two parameters and one taking three (non-defaulted) parameters. We make a corresponding change to Doubly_Linked_Lists.

Approve AI with changes: 13-0-0.

AI12-0401-1/01 Renaming of qualified expression of variable

Steve asks Tucker if he wants to show the example of case statement being out of range if this is not checked. It seems important to show the problems that can be caused. Tucker will provide this example to the editor.

Steve asks if there was a reason for not using “statically compatible” here? Tucker thinks that everything is compatible with the first subtype. Randy looks at the rules in 4.9.1 and does not see anything like that. Rather, the rule is both stricter and more relaxed than the rule proposed here; there’s nothing like the first subtype/base type part for “static compatibility”, while “static compatibility” does allow different subtypes with the same static constraint. We won’t make any change.

Richard notes that the subject does not read well:

> Renaming of {a} qualified expression of {a} variable

Gary sees a typo in the second paragraph of !discussion:

> … altered to no {longer} satisfy the nominal subtype of the qualified expression.

Capitalize the “if” in the question. “{Consider}[In]” at the start.

Gary asks if this is an incompatibility with Ada 2012. It is, and that should be mentioned in the !discussion.

Approve AI with changes: 13-0-1. John abstains.

AI12-0402-1/01 Master of a function call with no special result type

We want to allow local parameters to be passed to explicitly aliased parameters if it is blatantly obvious that an access to them cannot be buried in the result. If it could be hiding something that is build-in-place or some access type, there could be trouble (we're removing an existing guarantee for the latter).

Tucker wants the rule to be simple, so it might need to be inverted.

"Aurhor" should be "Author". "Aliasse". "Bullet" should be plural in the !discussion. "these paragraph{s}".

Approve intent: 7-1-6. Arnaud opposes, we are at the limit with the accessibility model; addressing this specific case isn't going to add anything but complexity. Bob, Ed, Gary, Claire, Jeff, John abstain.

There is an alternative approach of leaving the accessibility level definitions alone and just changing some (accessibility-related) legality rules. One could be a bit more aggressive that way (the build-in-place issues go away). Some people would prefer that approach.

AI12-0403-1/02 Presentation issues from Draft 26 review

Randy and Tucker explain the changes.

Let's remove (4) from this AI, as it needs more digestion. [It was placed into AI12-0404-1 after the meeting – Editor.]

Approve AI with changes: 14-0-0. (Without item (4)).

AI12-0405-1/00 Fixups for Stable Properties

[Editor's note: This AI number was assigned after the meeting.]

Should stable properties checks be restricted from **out** mode parameters? Tucker notes that one could require that something be stated about the stable properties of an **out** parameter.

Steve notes that would be essentially requiring the overriding rule to kick in for an **out** parameter. The idea is that stable properties are always well-defined.

Steve would prefer the simpler rule that one just doesn't say anything about **out** parameters. He doesn't like that a postcondition would be required and thus the absence of one would make the program illegal. Tucker doesn't buy this argument, it only happens when one defines a stable property, that makes a requirement. If you are defining stable properties, you are saying that you want postconditions to define them.

We seem to agree that the stable property check does not apply to **out** parameters. The other part is less clear.

Straw vote: In favor of requiring that the property appear in the postcondition for each out parameter. Randy, Tucker, Richard, Tullio, Brad; No extra requirement: Steve, Ed, John, Claire, Jean-Pierre, Arnaud. Abstain: Gary, Bob, Jeff.

This is not quite a tie, but it is as close to one as we can get without it being a tie. We'd like to see how complex this added rule really would be to help make a decision.

Tucker had made a suggestion for **in** parameters, but he's now convinced that it was a dubious idea, so we won't consider that.

Steve will take an action item to write an AI for stable properties, and he will write a trial version of the requirement to write a postcondition for **out** parameters.