Minutes of ARG Meeting 63E

18 July 2024

Attendees:

Steve Baird, John Barnes (leaves 11:05; returns 11:25), Randy Brukardt, Jeff Cousins (leaves 11:30, returns 12:50), Gary Dismukes (leaves 13:10), Bob Duff, Edward Fish, Jean-Pierre Rosen, Tucker Taft (leaves 13:10).

Observers:

Christoph Grein.

Meeting Summary

The meeting convened on Thursday, 18 May 2024 at 10:44 hours EDT and adjourned at 13:28 hours EDT. The meeting was held using Zoom. The meeting covered all of the new and updated AIs on the agenda. (Note: The agenda also contained two approved AIs [AI22-0102-1 and AI22-0107-1] due to a cut-and-paste error. These were skipped when the error was discovered.)]

AI Summary

The following AIs were approved:

AI22-0033-1/04 Additional terms and definitions (8-0-0)
AI22-0094-1/04 Assertion_Policy and preelaborability (9-0-0)
AI22-0104-1/02 Slicing of Vectors (8-0-0)
AI22-0106-1/03 Multi-dimension array_aggregates and iterators (8-0-0)
AI22-0109-1/01 Using class-wide types with user-defined literals (8-0-0)
AI22-0110-1/01 Disallow Put_Image aspect spces for interface types (8-0-0)
AI22-0111-1/02 Class-wide Container Aggregates (8-0-0)
AI22-0112-1/02 Assignment to generalized references with a target name symbol (9-0-0)

The following AIs were approved with editorial changes:

AI22-0097-1/04 Usage Advice (part 2) (8-0-1)
AI22-0108-1/01 Completion of Usage Advice (8-0-1)

The intentions of the following AIs were approved but they require a rewrite:

AI22-0022-1/03 Difficult examples issues from WG 9 review (7-0-0)
AI22-0101-1/02 Valid_Scalars attribute is defined (7-0-1)

Detailed Minutes

Welcome

Steve welcomes everyone to this meeting.

Apologies

Richard is in Hong Kong, the time difference makes attending the meeting impractical. Alejandro, Arnaud, and Justin all sent last minute apologies.

[Editor's Note: After the meeting, Niklas apologized for missing the meeting because his calendar program had somehow gotten the wrong date (thinking Friday was July 18th) and Brad also apologized as he was behind in his e-mail after vacation and missed the meeting announcement.]

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 9-0-0.

Date and Venue of the Next Meeting

Our next electronic meeting is proposed for Thursday, September 26 with the usual time (10:30-13:30 EDT [-4 UTC]) and method (Zoom). Several people have conflicts with that date. Jeff has a conflict with the previous week as well. The week after conflicts with a large AdaCore meeting. We look a week further out. WG 9 is meeting on Thursday that week, Jean-Pierre can't do Wednesday, Jeff can't do Tuesday. Jean-Pierre suggests we meet without him, so we settle on Wednesday, October 9th.

User Community Input Working Group Report

Richard is not here and no one knows of anything to report.

Future ARG Funding

Tullio is not here and we also know of nothing to report.

WG 9 Response (Possible Corrigendum)

Randy wrote and circulated a possible response to the WG 9 request. He recommends creating a Corrigendum soon, as the combination of oncoming ISO requirements and lack of funding makes future work problematic. A "checkpoint" now would be valuable (particularly if continuing proves impossible).

Randy had asked for Pat's input on this; he said he would respond "tomorrow" (this was on Monday); but apparently "tomorrow" never comes.

Ed asks what the cost for the commitment for the WG 9 convenor. It is hard to say, it is a three year commitment to fund travel and time and costs. For the US body, the fee is $1500/year for small organizations and more for larger ones. Other countries will vary.

We discuss what to do. Responding at the next meeting is too late; it would only be a few hours before the WG 9 meeting. It is suggested that we approve this response now and WG 9 can reject or modify it if necessary.

Approve response to WG 9: 9-0-0.

Unfinished Action Items

We did not discuss any of the unfinished action items.

Current Action Items

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

Steve Baird:

- AI22-0076-1 (with help from Tucker Taft)
- AI22-0101-1

Randy Brukardt:

- Create an AI to make the evaluation and Volatile semantics of the Valid (and Valid_Scalars) attribute normative (see discussion of AI22-0101-1).

Editorial changes only:

- AI22-0097-1
- AI22-0108-1

Jeff Cousins:

- Create an AI to define additional terms and definitions are suggested in the discussion section of AI22-0033-1 (and any others that seem necessary).

Edward Fish:

- AI22-0022-1

Christoph Grein:

- Create an AI with remaining issues split from AI22-0097-1.

Justin Squirek:

- ACATS C-Test(s) for filters

Tucker Taft:

- AI22-0063-1 (Split work with Richard Wai)
- AI22-0071-2
- AI22-0076-1 (assist Steve Baird)

Richard Wai:

- AI22-0063-1 (Split work with Tucker Taft)

Detailed Review

The minutes cover detailed review of Ada 2022 AIs (AI22s). The AI22s 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 202y AARM, the number refers to the text in draft 1 of the Ada 202y AARM. Paragraph numbers in other drafts may vary. Other paragraph numbers come from the final Ada 2022 AARM; again the paragraph numbers in the many drafts may vary.

Detailed Review of Ada 2022 AIs

AI22-0022-1/03 Difficult examples issues from WG 9 review

The Node_Map vector fix is missing, Ed thinks it was lost in editing.

Question (4) about a container iterator example was not handled yet.

Ed would like to add a section of large examples to Annex A. That would be new clause A.20, moving A.18.33 (which is a large example) to that clause as A.20.1, and this new example labeled A.18.34 would be A.20.2.

Randy notes that the examples in 9.1 are very simple constructs. We certainly don't want this one first (start with the simplest cases). So put it after 9.1(28) (keeping all of the types together and all of the objects together).

"ARG_Examples" is a bad name; just call that package "Examples".

No one had anything else, and our time is nearly up. [This was discussed last – Editor.]

Approve intent: 7-0-0.

AI22-0033-1/04 Additional terms and definitions

We look at the changes that Tucker made. These look good.

Jeff will write the definitions for the terms mentioned in the Future AIs part of the !discussion (since he's not on the call at the moment!), and create an AI to hold them.

Approve AI: 8-0-0.

AI22-0094-1/04 Assertion_Policy and preelaborability

The change from last time is that we except Restriction violations from the usual rule that Assertion_Policy does not affect Legality. Excepting subtype predicates, since those can never be turned completely off (they always are evaluated in 'Valid and memberships).

Ed asks why "and" in the existing wording of 13.12(9.1/1). It's existing wording; Steve notes that something affecting null statements could need "and" (they might be reachable and still have no code associated; the permission does not apply in that case).

Approve AI: 9-0-0.

AI22-0097-1/04 Usage Advice (part 2)

Tucker wonders what is the philosophy of those notes. Randy says we never wrote a mission statement for the "Usage" header (perhaps we should have). He thinks it was information that Ada users should have, as opposed to interesting corner cases or items most interesting for implementers or language lawyers.

With that in mind, we evaluate the notes that were considered questionable.

3.5.2(9/5) leave as a note (which means to delete it from the AI). Tucker notes that this was important in 1979, but it is primarily history today. It would make more sense to talk about Shift-JIS here, but that too is declining in importance and it would be hard to define in Ada as a user because of sheer size.

3.5.5(12/5) leave as note: it doesn't have much to do with usage.

3.5.10(16) Jean-Pierre says he thinks is should be Usage. Others concur. So remove the comment and leave it in the AI.

4.3.3(33/5) Tucker suggests moving the second sentence to the first, and then start the second sentence with "while". That avoids making a recommendation while making the advice clearer. The author's note should be removed or modified to not talk about rewording.

9.6.1(91/5) should be rewritten as a Usage advice with (some of) the AARM note included. So split this into a separate AI since rewording this here is going to take too long.

13.3(20) should be rewritten as Usage advice. Put that into the separate AI.

13.12(16/2) not really Usage, and there is no usage advice in either of the referenced clauses. So delete this from this AI.

13.13.1(38) put into new AI and fix examples.

B.3(75/4) Tucker thinks this is Usage, so leave this here.

B.3.1(58/5) is about implementation, not usage, so don't move it (and remove it from this AI).

D(3) About structure of RM, not usage. So remove this from this AI.

Christoph agrees to take the new AI with the handful of issues.

We give Christoph thanks for taking this on in the first place, it was a lot of work.

Approve AI with changes: 8-0-1. Jeff abstains as he just returned and didn't hear most of it.

AI22-0101-1/02 Valid_Scalars attribute is defined

Steve explains the AI.

He had some questions about corner cases.

In answer to his first question, Tucker notes that protected actions aren't potentially blocking (only entry calls block). They wait for exclusion, but that is not blocking.

Steve thinks that we don't want to talk about implementation-defined components. Tucker says we probably want to allow implementations to go further for things like the tag. Bob wonders why we need such a permission, he thinks that we're talking about erroneous objects. The whole point of this attribute (and Valid as well) is to avoid erroneous execution. If you use this attribute and get False, then so long as you don't touch the object further, you haven't become erroneous from this object.

We think you need to be able to return False even if the canonical semantics would say the result is True; the idea here is to prevent doing something that is erroneous. If the implementation can determine that the tag is invalid, it should be able to tell the user that, so they can avoid doing something that would trigger erroneous execution.

Steve wonders if we need a term like "underlying tag". We don't have such a term, and it is sometimes useful. It is not needed at the moment, and we don't know of other uses that could use the term.

Randy had noticed that Note 2 seems like it needs to be normative, not a note. Especially the treatment of volatile objects; there doesn't seem to be any way to derive this semantics. Randy will take an action item to write an AI to make that normative.

Approve intent: 7-0-1 (John abstains).

AI22-0104-1/02 Slicing of Vectors

Tucker fixed up the Pre and Post on these operations. There are other changes he would like to make, but that clearly would be another AI.

Steve asks if it is clear that the Replace_Slice procedure copies the data of the Source. We think that is assumed, since there are a few list operations that say they *don't* copy the data. That wouldn't be necessary if the model wasn't that copies are always done.

Ed asks about restructuring of these packages to avoid some of the duplication of contracts. We think that is very hard to do compatibly (extra instances are necessary in most cases). In any case, it is a different question; a proposal is welcome on the Github.

Approve AI: 8-0-0.

AI22-0106-1/03 Multi-dimension array_aggregates and iterators

Randy explains the AI and the changes. The main change was to clarify the wording so it could not be construed to apply to aggregates that are used as components of an enclosing aggregate. Unfortunately, both the entire aggregate and component associations include the component expressions, so we have to be even more specific (which leads to wording that looks a bit redundant).

Approve AI: 8-0-0.

AI22-0108-1/01 Completion of Usage Advice

Randy suggests the subject and summary should reflect the actual change. Change to: "Usage Advice about File Renaming".

Approve AI with changes: 8-0-1 (Ed Fish abstains).

AI22-0109-1/01 Using class-wide types with user-defined literals

It is odd that you can call the underlying function with a class-wide object, but not using the syntax sugar. In the example in the !issue, consider that:

```
X := From_String ("123"); -- OK.
X := 123; -- Illegal with current rules.
```

The second expression is equivalent to the first, but the legality is different.

Tucker explains that it turns out that only the Name Resolution rules need changes in this case, all of the other requirements are covered by the "nonoverridable" rules.

John asks who is George in the example. Tucker explains that it is sort of George Washington, but he invented the biographical details rather than looking them up.

Approve AI: 8-0-0.

AI22-0110-1/01 Disallow Put_Image aspect spces for interface types

Steve explains the AI. Put_Image of an interface can only put out a constant string or something depending on globals. Tucker notes that if it is primitive, it can only be null, which is even less useful.

The change is technically incompatible, but GNAT crashes when this is tried, so no one has ever done this before Steve.

Approve AI: 8-0-0.

AI22-0111-1/02 Class-wide Container Aggregates

This is the same idea as AI22-0109-1. Ed asks if that would allow an empty set to be defined for a set interface. Yes, that is the sort of use that is envisioned; we even go out of our way to allow the routines to be abstract for that reason.

Steve wonders if this adequately defines what happens in all of the possible cases. Tucker says that the 3.9.2 rules apply in all of other cases. Randy notes that the idea is that this works very much like user-defined literals. If you have X = […], you want the tag of the aggregate to be that of X, and the 3.9.2 rules do that.

Tucker starts looking at the 3.9.2 rules, but Randy notes that those were rewritten recently [by AI22-0043-1 - Editor], and he shouldn't look to closely at them lest Steve find some problem we've already fixed.

Approve AI: 8-0-0.

AI22-0112-1/02 Assignment to generalized references with a target name symbol

Tucker explains that Ada 2005 changed assignment of nonlimited objects from a resolution rule to a legality rule. It was solely a consistency issue, but it is causing ambiguity problems for target name symbols.

The name of an object of a reference type always has two interpretations. An assignment containing a target name symbol means that the target is a complete context. If there are two interpretations of the target, then the assignment is illegal as the target is ambiguous. By ignoring the interpretation that is of a (limited) reference object, we side-step the problem.

This is important as variable_indexings are required to return reference objects. So if V is a vector, something as simple as:

```
V(1) := @ + 1;
```

is always illegal with the current rules (even though the interpretation of the target V(1) as a reference object would be an illegal assignment).

If someone ignores the advice to make a reference type limited, a target name assignment is still ambiguous. The assignment that explicitly writes out all of the operands would still work. We could fix that with a lot more complexity, but it doesn't seem worth it today. This change would not prevent fixing it "properly" in the future (such a fix would be an extension).

Approve AI: 9-0-0.