Minutes of ARG Meeting 63D

2 May 2024

Attendees:

Steve Baird, John Barnes (leaves 13:04), Randy Brukardt, Jeff Cousins, Gary Dismukes, Bob Duff, Edward Fish, Niklas Holsti, Brad Moore, Jean-Pierre Rosen, Justin Squirek, Tucker Taft, Tullio Vardanega (joins 11:03), Richard Wai.

Observers:

Mohammad Ali Asgar (Canada), Christoph Grein (Germany), Robin Leroy (Unicode liaison)

Meeting Summary

The meeting convened on Thursday, 2 May 2024 at 10:39 hours EDT and adjourned at 13:28 hours EDT. The meeting was held using Zoom. The meeting covered many of the AIs on the agenda.

AI Summary

The following AIs were approved:

AI22-0091-1/03 Generalize prefixed views (12-1-1)
AI22-0096-1/02 Ada and OpenMP (14-0-0)
AI22-0098-1/04 More presentation fixes (14-0-0)
AI22-0099-1/02 Object_Size and conversions of access-to-object types (13-0-1)
AI22-0105-1/02 Meaning of conforms to this Reference Manual (12-0-2)

The following AIs were approved with editorial changes:

AI22-0100-2/02 Upper bound calculation (14-0-0)
AI22-0102-1/02 Substitute parameter for UTF Decode functions (14-0-0)
AI22-0103-1/01 Double iteration for array aggregates should only evaluate iterators once (14-0-0)
AI22-0107-1/02 Specifying the Storage_Size for the environment task (13-0-0)

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

AI22-0104-1/02 Slicing of vectors (14-0-0)
AI22-0106-1/02 Multi-dimension array aggregates and iterators (14-0-0)

The following AIs were discussed and assigned to an editor:

AI22-0094-1/02 Assertion policy and preelaborability (13-0-1)

The following AIs were discussed and voted No Action:

AI22-0100-1/02 Bounds of null and positional array aggregates (14-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone to this meeting.

Apologies

Alejandro can’t attend.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 13-0-0.

Date and Venue of the Next Meeting

Our next electronic meeting is proposed for Thursday, July 11 with the usual time (10:30-13:30 EDT [-4 UTC]) and method (Zoom). Tucker has a conflict, so we move the date to Thursday, July 18th.

User Community Input Working Group Report

Richard says that he finally did update the Readme and removed some confusing files.

Future ARG Funding

We have not heard from Tullio, and he's not here.

Later, after Tullio arrived, we asked him for a status update.

He has created a new charter document for the ARG after discussing with AdaCore management. It will be submitted to WG 9 to discuss.

Ada-Europe members did not approve the conversion to Ada User Society. So what the successor organization will be is in limbo.

Unfinished Action Items

Randy notes that Alejandro sent us a mail saying he couldn't participate for a while. He had asked whether we should reassign his homework, but never received any answer to that message. The management will take an action item to decide what to do with Alejandro's homework.

Current Action Items

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

Steve Baird:

- Decide, with Randy, how to handle Alejandro's homework.
- AI22-0076-1 (with help from Tucker Taft)
- AI22-0094-1

Randy Brukardt:

- Decide, with Steve, how to handle Alejandro's homework.
- AI22-0106-1

Editorial changes only:

- AI22-0100-2
- AI22-0102-1
- AI22-0103-1
- AI22-0107-1

Edward Fish:

- AI22-0022-1

Alejandro Mosteo:

- Github Issue #5 (with help from Tucker Taft) – to be reassigned
- Github Issue #61 (with help from Tucker Taft) – to be reassigned

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)
- AI22-0104-1
- Github Issue #5 (assist Alejandro Mosteo as needed)
- Github Issue #61 (assist Alejandro Mosteo as needed)

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

AI22-0091-1/03 Generalize prefixed views

Gary notes the changes to the AI.

Jeff asks if this has been prototyped – Gary answers that it has, in GNAT as an extension feature.

Approve AI: 12-1-1. Ed opposes – he doesn't think that prefixed notation is worth the complexity to add to all types. Jean-Pierre abstains, he says he basically concurs with Ed.

AI22-0094-1/02 Assertion policy and preelaborability

Should Restrictions be covered in this AI? Tucker notes that there is a customer that wants to be able to use recursion in assertions but not in code. It is noted that the recursion restriction, which is not enforced, is not the best example of this problem, but it seems that a similar problem could occur for any restriction.

Randy notes that a conditional compilation scheme could deal with that. Steve notes that one is already changing their assertion policy pragma; they could also do the same thing with the restrictions pragmas. So they could change both of the pragmas at the same time.

Niklas notes that the assertions would still remain, and they would still be enforced against the restrictions pragma. So when the target is compiled, the restrictions are going to be violated (if legality is still enforced on assertion expressions).

So we do have a problem.

Tucker argues that Restrictions are a funny thing that should be handled differently.

So we need to do something. Should Restrictions be handled in a separate AI? Tucker notes that the user note explaining the design principle would need to be changed if restrictions are changed. That argues for a single AI.

Randy does a 180. He wonders if we want the principle at all. He notes that static expression legality rules cause all kinds of problems with conditionally compiled code. This seems to be the same sort of issue – code that can never be executed making a program illegal.

[Editor's note: For those that are new to this issue, there is a rule that a static expression that raises an exception is illegal – 4.9(34/3). Ada doesn't have a separate conditional compilation feature, rather one uses if statements with appropriate static conditions. If a static expression appears in code surrounded by an if statement that starts with (after evaluation) `if False …`, and that static expression would raise an exception (for instance, if a constant is zero and it is the divisor in a static expression), then the code is illegal even though it could never be executed (and it would be legal in any case where it could be executed.) That can cause problems late in development when tracing facilities are turned off. There are workarounds but they are a pain and possibly impact performance.]

Tucker disagrees that the principle is bad. He does not want to throw out the baby with the bathwater. [Randy is unsure that there is a baby in the bathwater, it might just be some dirty towels.]

Steve will take this back and reconsider the rules.

Keep alive: 13-0-1.

AI22-0096-1/02 Ada and OpenMP

Tucker added an AARM note as requested, and a few other changes associated with that. But no "meat" changes.

Approve AI: 14-0-0.

AI22-0098-1/04 More presentation fixes

Randy explains the fixes. Three of these problems showed up when he was putting rules into the RM draft, and the other two were reported by users on Github. He notes that changing an example is non-normative, so it can be considered presentation in a case like there where there is no functional change.

Approve AI: 14-0-0.

AI22-0099-1/02 Object_Size and conversions of access-to-object types

Randy explains the AI. We cannot allow an access value to designate an object that is not the size that it expects. He notes that compilers are not required to support specifying non-confirming Object_Sizes for record types, but we have to define the language so it makes sense if they do support such specifications.

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

AI22-0100-1/02 Bounds of null and positional array aggregates

This was an initial attempt at AI22-0100-2. It's more complicated, and would not work well for concatenation.

No action: 14-0-0.

AI22-0100-2/02 Upper bound calculation

Randy explains the AI in detail.

The 3.6.1(8) AARM Reason has a trailing curly bracket, it should have a closing parenthesis inside of the period. There are some junk tabs in the AARM note.

Randy notes that is would be nice if some ACATS tests appeared for these cases that don't already have tests. Steve notes that Randy is talking to Jeff when he says that.

Approve AI with changes: 14-0-0.

AI22-0102-1/02 Substitute parameter for UTF Decode functions

Robin notes that Unicode has a recommended substitution character, so he is happy that there is no default here. Randy jokingly suggests that he not open the RM to Ada.Characters.Conversions. It uses ' ' as the default substitution character; that dates back to Ada 95 (before there was any serious consideration of Unicode). Robin agrees that it is too late to change that.

Robin asks what happens for a malformed input string. Randy thinks that a malformed input string will still raise Encoding_Error. Tucker concurs. We note that A.4.11(53-54) note that malformed input always raises Encoding_Error. Robin notes that Unicode has a strategy for handling such strings. We're dubious that that is helpful.

Robin complains about the fact that an overlong encoding is not considered invalid by this rule (A.4.11(53-54)). That was an explicit decision taken in AI12-0088-1.

Both of these things are separate problems; Robin can make a proposal if he feels strongly enough.

Tucker aligned the colons in some of the declarations.

Approve AI with changes: 14-0-0.

AI22-0103-1/01 Double iteration for array aggregates should only evaluate iterators once

Steve notes that the Google Docs file name is missing a space in the subject. Randy takes the opportunity to remind everyone that subjects need to be kept short enough to fit on one line in the Google Docs index, in agendas, and in editorial review lists. (Also, WG 9 passed a resolution mandating that at one point.) That means that should not exceed about 50 characters. This one is too long.

Shorten the title to "Array aggregate double iteration".

Tucker explains the AI. The rules have the effect of evaluating the names twice, which might give different results from each evaluation for some possible iterators. We need the two iterations to be as similar as possible.

We need to add access_definition to the elaboration, as it is in the syntax and it is bizarre not to elaborate it.

Approve AI with changes: 14-0-0.

[Editor's note: The HTML version has an error in the numbering of the choices that should be fixed if possible.]

AI22-0104-1/02 Slicing of vectors

Tucker explains the AI.

Tucker notes that he did not allow super-null slices. That is inconsistent with the Unbounded Strings case, which only requires that the bounds are in range. Richard does not like that inconsistency.

Tucker doesn't like having to specify a capacity check on each subprogram. He'd rather have that as a type invariant or stable property. Randy notes that is only necessary for operations that might expand the length, the majority of operations do not do that (including operations like Delete). If one makes it apply to all operations, then it will be checked by many operations that cannot fail. Tucker says that compilers could optimize it away. [That's true, but a compiler would have to prove three things: that it was true when the subprogram was entered, that that capacity

didn't change, and that the length didn't grow. It's likely that those would be hard for most procedures, especially ones that change but only shrink the length.]

Niklas had noted that there needs to be a tampering check in the preconditions of the ones that are modifying a container. Randy notes that there is no longer a list of places that require a tampering check; those are given in the preconditions.

Brad says that he has a package like this with Delete_Slice and Find_Slice. Delete_Slice can be handled by using Delete with a position and a count. Find_Slice seems like a different thing, it is suggested that he propose a separate AI for that.

Should this be a Binding Interpretation? It's not fixing a bug, and compilers have mechanisms for handling such additions. There doesn't seem to be any critical reason for allowing it in earlier language versions.

Approve intent: 14-0-0.

AI22-0105-1/02 Meaning of conforms to this Reference Manual

Randy explains that "conforms to the RM" and "conforms to the core" meant different things, which was confusing.

Tucker adjusted the wording to talk about "fully" conforms to the core, and then adjusted the other paragraph so it always applies.

Approve AI: 12-0-2. Steve and Gary abstain.

AI22-0106-1/02 Multi-dimension array aggregates and iterators

Randy tries to explain the AI.

There is a missing phrase in the !discussion (Tucker fixes).

The !issue should mention the issue with index parameters of discrete_ranges.

Steve complains that "shall not appear" includes subexpressions, and we don't want to be saying anything about aggregates that happen to be inside of the aggregate. So that wording needs revisions. It's not immediately clear how to reword this to include the aggregate and its subaggregates, but not any aggregates used in component expressions given in the aggregate.

The "shall not appear" is OK for the second paragraph, since it is very specific, and we really don't want any uses anywhere in any subexpression of the discrete_range_list.

Randy will take this back and make the needed fixes.

Approve intent: 14-0-0.

AI22-0107-1/02 Specifying the Storage_Size for the environment task

Richard is uncomfortable with the notion that specifying the properties of the environment task associated with the main subprogram. He notes that a partition doesn't need to have a main subprogram at all.

Randy notes that Priority and CPU have always worked this way. Priority has been this way for 30+ years, and the inability to specify the priority of the environment task of a partition without a main subprogram has been a problem all of that time, and no one has been demanding a change.

Bob notes that while there is a linker option for setting the stack size in GNAT, there's no way to do so in the language.

Randy thinks it is important to treat this the same way as Priority and CPU. We want these things to be consistent. And there is no chance that we would remove that existing capability. So we need this AI in any case. Richard agrees that if we already have this mechanism in other cases, it makes sense to do it here as well.

It is suggested that a proposal for another way to specify this (and Priority and CPU) on the environment task is a separate AI.

Gary notes that the insertion brackets are missing in 65.1/3.

Approve AI with changes: 13-0-0.