Minutes of ARG Meeting 63C

22 February 2024

Attendees:

Steve Baird, John Barnes (leaves 13:12), Randy Brukardt, Jeff Cousins, Gary Dismukes (leaves 12:42), Bob Duff (leaves 12:27), Edward Fish, Niklas Holsti, Brad Moore (leaves 12:01, returned about 13:12), Alejandro Mosteo, Jean-Pierre Rosen, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai .

Observers:

Mohammad Ali Asgar (Canada), Robin Leroy (Unicode liaison, leaves at 13:00)

Meeting Summary

The meeting convened on Thursday, 22 February 2024 at 10:37 hours EST and adjourned at 13:15 hours EST. The meeting was held using Zoom. The meeting covered all of the AIs and a few of the Github Issues.

AI Summary

The following AIs were approved:

AI22-0059-1/02 Parallel_Calls aspect for types (8-0-4)
AI22-0086-1/03 Conversions of general access-to-object values (14-0-1)

The following AIs were approved with editorial changes:

AI22-0055-1/04 Usage Advice (15-0-0)
AI22-0083-1/04 Treat dynamically-tagged expressions as class-wide in various contexts (12-0-3)
AI22-0092-1/01 Recommend limited reference types (15-0-0)
AI22-0093-1/01 No_Return glitches (15-0-0)
AI22-0095-1/01 The number of values of predefined character types (15-0-0)

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

AI22-0022-1/02 Difficult example issues from WG 9 reviews (15-0-0)
AI22-0091-1/01 Generalize prefix views (12-1-0)
AI22-0094-1/01 Assertion_Policy and preelaborability (15-0-0)
AI22-0096-1/01 Ada and OpenMP (13-0-1)

The following Github Issues were discussed and assigned to an editor to create an AI:

#14 Valid_Scalars attribute worth standardizing
#73 Specify Storage_Size for environment task
#75 Add slicing for Ada.Containers.Vectors
#78 Substitute parameter for Decode for UTF strings

The following Github Issues were discussed and voted No Action:

#77 Creation from array and replication of Vectors (12-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone to this meeting.

Apologies

Gary said he'll have to leave early. Bob and Brad say the same. Arnaud sent a message at 10:30 saying he cannot attend.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 15-0-0.

Date and Venue of the Next Meeting

Our next electronic meeting is proposed for Thursday, May 2 with the usual time (10:30-13:30 EDT [-4 UTC]) and method (Zoom). There are no objections to this date.

User Community Input Working Group Report

Richard says there is nothing new to report. He still is planning to update the Github Readme.

Future ARG Funding

Randy reminds everyone that he has been informed by AdaCore that they will fund 2024 as usual, but do not intend to fund the ARG in future years (2025 and later). Much has been discussed, but no decisions have been made at this point.

Tullio notes that the Ada User Society (the union of Ada Europe and SIGAda) is working on taking over this work. Much of the effort to this point has been focused on setting up the Ada User Society itself.

Unfinished Action Items

Randy would like to finish versions of AI22-0033-1, AI22-0055-1, and Alejandro's AIs, so that they can be added to the RM while we still have funding. For the two existing AIs, we should split them into a part that is finished and a part for future work (we had already decided to do that with AI22-0033-1). Alejandro promises to do some work for next time.

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 assistance from Tucker Taft)
- AI22-0094-1
- Github Issue #14

Randy Brukardt:

- Github Issue #78

Editorial changes only:

- AI22-0055-1
- AI22-0083-1
- AI22-0092-1
- AI22-0093-1
- AI22-0095-1

Gary Dismukes:

- AI22-0091-1

Edward Fish:

- AI22-0022-1
- Github Issue #73

Alejandro Mosteo:

- Github Issue #5 (with help from Tucker Taft)
- Github Issue #61 (with help from Tucker Taft)

Justin Squirek:

- ACATS C-Test(s) for filters

Tucker Taft:

- Set up resources and recruit people to help prototype new features (both Ada 2022 and new ones) in various open source technologies.
- OpenMP Technical Report.
- AI22-0033-1
- AI22-0063-1 (Split work with Richard Wai)
- AI22-0071-2
- AI22-0076-1 (assist Steve Baird)
- AI22-0096-1
- AI22-0097-1 (see discussion of AI22-0055-1)
- Github Issue #5 (assist Alejandro Mosteo as needed)
- Github Issue #61 (assist Alejandro Mosteo as needed)
- Github Issue #75

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

AI22-0022-1/02 Difficult example issues from WG 9 reviews

The AI itself wasn’t updated. Ed sent examples via the ARG list. The examples have what goes in each place marked.

Randy notes that examples don’t allow forward references.

This needs to be put into the form of an AI.

Approve intent: 15-0-0.

AI22-0055-1/04 Usage Advice

Randy suggests splitting this. The wording proposed in this AI should be handled now while we have a proper ARG budget. We then should create a second AI to handle additional changes (and give that back to Tucker). [Editor's note: This AI has been assigned number AI22-0097-1 and has been assigned to Tucker.] Randy notes that he’s already added the Usage category to the RM tools.

Jeff asks whether the Usage sections would appear in the ISO version. We are not allowed to use words like “should” in Notes, but it is OK to use such wording in the main part of the Standard. So it makes the most sense to have a category for usage recommendations, as we don’t want to mix up these recommendations with ones for implementers. When we do that, then they can appear in the ISO version as well as in the RM.

We didn’t do that for the ISO standard for Ada 2022, as it felt like adding a new category after all of the approvals have been finished was too much change. So the usage notes were omitted from the Ada 2022 ISO standard. But it’s certainly fine to include them going forward so long as they are not marked as “Notes”.

The !discussion section should just talk about what this AI does and not any other ideas (those can go into the new AI).

No one has any improvements to the wording as given in the AI. So we can vote on this one (the exact wording of the !discussion not being critical).

Approve AI with changes: 15-0-0.

AI22-0059-1/02 Parallel_Calls aspect for types

Tucker explains the changes.

Jean-Pierre asks what happens if the aspect is applied to a subprogram that is not callable in parallel. Tucker notes that the conflict checks are applied to the subprogram. So the aspect provides a “boundary”; outer calls can assume the subprogram is OK when doing conflict checks; that is verified inside the subprogram. Randy notes that if conflict checks are off, it doesn’t have any effect.

Jeff asks whether this needs prototyping. Tucker notes that this is building on conflict checking – which already exists without prototyping. Tucker thinks it is a fix to conflict checking – and thus it doesn’t need prototyping separately. No one disagrees with this intent.

Steve asks about whether Image and Put_Image need to be covered. Randy thinks that separate calls usually have separate buffer objects, so there is no parallelism problem. (And if an implementation is using a single shared buffer, it is going to have problems meeting the basic requirements of the Ada language.) Additionally, all operations, implicit or explicit is covered by the definition of the conflict checks, so they will be taking into account if necessary.

Approve AI: 8-0-4. Steve, Justin, Ed, John abstain.

AI22-0083-1/04 Treat dynamically-tagged expressions as class-wide in various contexts

Steve wonders if “root type” is well defined. After research, we add “of the class” to the text.

Ed asks a question about the example; he’s worried about a silent subsetting of the type. In the renaming case, there is no subset (or conversion). If the Union call has been on objects of type Set’Class, then the renaming would be illegal.

Steve wonders if this would require a lot of work for compilers. We don’t think so, because this is allowed in other contexts. Randy notes that we can never tell for sure what is hard for implementations. But there is nothing new dynamically here (which is the hard part of implementing tag-indeterminate calls), this is just statically allowing some additional cases.

Approve AI with changes: 12-0-3. Jeff, Justin, John abstain. [Editor's note: This AI was just voted to "Approve AI", but it should have been voted "with changes" as a change in the wording is noted in the text above. I've corrected the type of vote here. No ARG members objected to this change when asked.]

AI22-0086-1/03 Conversions of general access-to-object values

Steve notes that there is another AI in the pipeline to handle Object_Size.

Approve AI: 14-0-1. Gary abstained.

AI22-0091-1/01 Generalize prefix views

We discuss the AI. Randy suggests that we have an AARM note to clarify that visible access types cannot use this notation because of the automatic implicit dereference.

This is implemented in GNAT as an extension. So no prototyping is needed.

Randy is dubious that a preference rule would really work without introducing silent changes of meaning for some calls (something we never allow because of the danger). Preference rules have failed in many other resolution circumstances. But since we're not trying to do that, we can leave that as a question not answered.

Niklas notes that the example has problems. He is directed to send Gary an e-mail.

Approve intent: 12-1-0. Oppose: Ed Fish.

Ed Fish does not like this feature as defined. He is not a fan of the tagged version, either. He would like an aspect to allow other parameters to be the designated one, but of course that is a larger change in many ways.

AI22-0092-1/01 Recommend limited reference types

Tucker asks that the example be given as Ada code, following the proposed wording.

Approve AI with changes: 15-0-0.

AI22-0093-1/01 No_Return glitches

The crux of the problem here is that a null_procedure_declaration doesn't necessarily declare a null procedure; it also can be a completion. Randy notes that the syntactic entity is misnamed, it is not always a declaration. For subprograms, we allow subprogram_body to be a declaration or a completion, and we probably should have named these entities similarly. But it is too late for that, the amount of change would be substantial and the value would be relatively low.

Tucker prefers "defined by" rather than "declared by". A completion doesn't declare anything. Steve suggests "defined by or completing", which just seems wordier.

Ed asks why we don't allow instances to be specified. The meaning is checked on the generic unit, and it is not possible to change it in an instance. Steve suggests adding "subprogram" to the generic wording, since we're only talking about subprograms (not packages).

Approve AI with changes: 15-0-0.

AI22-0094-1/01 Assertion_Policy and preelaborability

[Editor's note: This AI was given version /00, but that is appropriate only for empty, unwritten AIs. So it has been corrected to version /01.]

Steve notes that we add a rule to ensure that changing Assertion_Policy does not change the legality of a program because of preelaboration. We add a note to state the principle in 11.4.2.

The !discussion point needs to be answered. Tucker notes that this came up in customer code: there was a restriction No_Recursion and a pragma Assert that makes a recursive call. The restriction should apply regardless of whether the policy ignores the pragma.

The note is not intended to be normative, so it seems that we need a rule somewhere (probably 13.12 since it should apply to all restrictions) to say that Restrictions apply regardless Assertion_Policy. Tucker suggests probably in the Post-Compilation Rules.

So the AI goes back to Steve. He should also improve the !discussion.

Approve intent: 15-0-0.

AI22-0095-1/01 The number of values of predefined character types

Tucker notes that there is 0 included, so the number should be 11141112, 0 .. 16#10FFFF#. Steve complains that we want to keep the wording similar between the three cases. Robin concurs about using a range. "position numbers in the range 0 .. 16#10FFFF#" or something like that.

We discuss whether 2**32 is preferable to the long number starting with a 4.

Jean-Pierre wonders why we even allow values out of range. That would change the behavior of existing programs, the inconsistency doesn't seem justified. Robin notes that Wide_Wide_Character corresponds to UTF-32, which uses 32-bits even though only 21 are actually used.

Tucker inserts all of the edits. Jeff suggests dropping "which" and the extra "values".

Approve AI with changes: 15-0-0.

AI22-0096-1/01 Ada and OpenMP

Randy asks Tucker what he is recommending we do with this. This is more of a technical report than an AI. Since ISO has various rules about what can or cannot be in Technical Report, so we probably shouldn't try to make this an official document.

John asks how it relates to parallel features. Tucker notes that it defines expansions. More generally, it provides a way to implement the parallel features in a compiler using OpenMP.

Tucker suggests an AARM implementation note to point at this AI. That seems useful, and it makes more sense than copying the entire thing into the AARM.

Ed notes that the other parallel are OpenCL and CUDA. OpenMP now covers everything in OpenCL, and CUDA is specific to Nvidia (and can be used to implement OpenMP).

Tucker is directed to craft an AARM note to point at this AI, and write this AI as a Ramification with a !wording section as the AARM note.

Approve intent: 13-0-1. Jean-Pierre abstains.

Detailed Review of Github Issues

#14 Valid_Scalars attribute worth standardizing

Steve will take this one.

#73 Specify Storage_Size for environment task

Storage_Size is not allowed to be specified on the main subprogram. Aspect Priority has rules that allow it to be specified on a subprogram with additional limitations so that it only applies to the main subprogram. Aspect Storage_Size needs similar rules in order to do this.

Several people think this is already possible, but upon reflection that's because of implementation-defined pragmas or project options. This capability should be provided in a standard way.

Ed Fish will take this issue.

#75 Add slicing for Ada.Containers.Vectors

This seems useful and relatively easy. Tucker will take this one.

#77 Creation from array and replication of Vectors

Randy notes that To_Vector and To_Array can be done with aggregate iterators. And the type cannot be written for indefinite containers.

The "*" operator isn't obviously useful for Vectors, certainly not in the way it is for strings. And at least some members don't find it a natural way to describe an array operation. For small N, it can be written as an aggregate, or use the "&" operator.

[After the meeting, Tucker noted that the "*" operation can be written as:

```
        [for I in 1 .. N => Vec]'Reduce("&", [])
```

This is not the most intuitive expression, but it does show that the "*" function can already be written by a user.]

Randy will write a rationale and close the issue.

No Action: 12-0-0.

#78 Substitute parameter for Decode for UTF strings

Randy notes that doing this in two steps is inefficient and error-prone. Randy will take this one.