Minutes of ARG Meeting 63B

14 December 2023

Attendees:

Steve Baird, John Barnes (leaves 13:16 EST), Randy Brukardt, Arnaud Charlet (leaves 11:25 EST), Jeff Cousins, Gary Dismukes, Bob Duff, Niklas Holsti, Brad Moore, Justin Squirek (leaves 12:52 EST), Tucker Taft, Richard Wai.

Observers:

Mohammad Ali Asgar (Canada; leaves 12:48 EST), Ed Schonberg.

Meeting Summary

The meeting convened on Thursday, 14 October 2023 at 10:53 hours EST and adjourned at 13:24 hours EST. The meeting was held using Zoom. The meeting covered all of the regular AIs and many of the amendment AIs.

AI Summary

The following AIs were approved:

AI22-0051-2/02 Preelaborable_Initialization and contract aspects (11-0-1)

The following AIs were approved with editorial changes:

AI22-0082-1/02 Problems with nonlimited reference types (11-0-1)
AI22-0085-1/02 Various presentation issues (11-0-0)
AI22-0087-1/01 Text buffers are initially empty (10-0-1)
AI22-0088-1/01 Privacy and Put_Image (11-0-0)
AI22-0089-1/02 Current Instance for singleton task/protected objects (11-0-0)
AI22-0090-1/02 Allow No_Return aspect on formal subprograms (10-0-0)

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

AI22-0083-1/03 Treat dynamically-tagged expressions as class-wide in various contexts (11-0-0)
AI22-0086-1/02 Conversions of general access-to-object values (11-0-0)

The following AIs were discussed and assigned to an editor:

AI22-0071-2/01 Provide subtype attribute for testing assignability and declaring a copy

The following AIs were discussed and voted No Action:

AI22-0051-1/04 Preelaborable_Initialization and contract aspects (12-0-0)
AI22-0077-1/01 Discussion on Accessibility Checking (9-0-2)

Detailed Minutes

Delay

The meeting start was delayed when Randy couldn't hear the meeting from his computer. Everyone could hear him swearing at his computer. Eventually, he called in to get working audio. During the break, he finally found the Zoom settings menu and found that the volume had somehow been set to minimum (it appears to be separate from the system volume setting).

Welcome

Steve welcomes everyone to this meeting.

Apologies

Jean Pierre said he'd be late or not come at all. Alejandro said he has a conflict. Arnaud will have to leave early. Tullio also has a last minute conflict and will not be able to attend.

Previous Meeting Minutes

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

Date and Venue of the Next Meeting

Our next electronic meeting is proposed for Thursday, February 22 with the usual time (10:30-13:30 EST [-5 UTC]) and method (Zoom). There are no objections to this date.

We are not planning a face-to-face meeting this year (specifically, we won't meet in Barcelona after the Ada-Europe conference). We do not think that we will have enough work to meeting the criteria for such a meeting. We'll have a virtual meeting as usual.

ARG Procedures Update

Randy notes that he was given an action item to update the procedures to take into account the possibility of virtual meetings (like this one) and to add the criteria for choosing between face-to-face and virtual meetings. He circulated the proposed updates before the meeting.

No one has any comments on the proposed updates.

Approve revised ARG procedures: 12-0-0.

User Community Input Working Group Report

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

Future ARG Funding

Randy says 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). We've started discussing possible funding for future years and associated changes. No decisions have been made at this point.

Tucker notes that Ada-Europe is considering broadening their mission beyond just Europe, and that could include continuing standardization activities. Unfortunately, Tullio wasn't able to attend this meeting to say more.

Unfinished Action Items

We decided not to discuss the unfinished action item list. Randy notes that he heard from several people with apologies for not making progress on their 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 assistance from Tucker Taft)
- AI22-0086-1
- Research Assertion_Policy and preelaboration (see discussion of AI22-0051-2).
- Define Character'Last and Wide_Character'Last (see discussion of AI22-0085-1).

Randy Brukardt:

Editorial changes only:

- AI22-0082-1
- AI22-0085-1
- AI22-0087-1
- AI22-0089-1
- AI22-0090-1

Edward Fish:

- AI22-0022-1

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
- AI22-0055-1
- AI22-0063-1 (Split work with Richard Wai)
- AI22-0071-2
- AI22-0076-1 (assist Steve Baird)
- AI22-0083-1
- 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

AI22-0051-1/04 Preelaborable_Initialization and contract aspects

This has been replaced by the other alternative.

No Action: 12-0-0.

AI22-0051-2/02 Preelaborable_Initialization and contract aspects

Randy explains the suggested rules. We need to be able to make certain contract checks in preelaborated code. This requires calling (usually) simple functions during preelaboration. We also don't want to break the basic properties of preelaborated units. The important one here is that no preelaborated code can need an elaboration check; it is required by C.4 that no code be generated for elaboration checks.

That is usually ensured by not allowing any functions to be called in preelaborated code, but as previously noted, calling functions is a necessary part of evaluating contracts. Thus, the idea of these rules is to ensure the functions needed to make allowed contract checks can be called without failing an elaboration check.

Randy says that he crafted these rules so that restrictions are only applied to types that explicitly specify Preelaborable_Initialization. If a type has Preelaborable_Initialization explicitly declared, it ought to be preelaborable. On the other hand, for a type that automatically has preelaborable initialization (like an elementary type), we don't want to restrict the contracts since no one might try to use the type in a preelaborated package (restrictions posing a possible compatibility problem). In such a case, the type might not be usable to declare a preelaborated object.

Randy also noted that there is no important reason to allow checks to be deferred; the implementation would be fairly complicated (involving the binder).

Tucker notes that he had forgotten that preelaborated packages still have to perform constraint checks during elaboration. So performing contract checks is pretty normal (so long as it doesn't break the assumptions surrounding preelaboration).

Steve notes that the rules for preelaboration are based on what will be executed, so he wonders if the assertion policy matters. Definitely not for these rules, these are based on the presence and contents of contracts, not whether they will be executed.

There is a question as to whether the assertion policy matters for the case where it is legal based on these rules. For example, for a type which has preelaborable initialization by default (such as an elementary), there is a Dynamic_Predicate that calls a function and applies to a subtype, and the subtype is used to declare a library-level object in a preelaborated package. In this case, if the assertion policy is Check, the object is certainly not preelaborable (it calls a function), but if the assertion policy is Ignore, is that still true? Steve will take an action item to research this.

Approve AI: 11-0-1 Brad abstains.

AI22-0071-2/01 Provide subtype attribute for testing assignability and declaring a copy

Tucker has proposed an attribute that can be applied to any object to get its actual subtype. (Not the nominal subtype.)

Steve notes that this would require changes to the accessibility rules. He points out that the accessibility of an object matters for access discriminants, and this would provide a way to move that accessibility to some other scope.

Steve also notes that we would not statically know whether or not such a subtype is statically constrained. Properties associated with the nominal subtype would be unknown in general (certainly for generic formal types).

Steve says that it would be OK if it is defined by an equivalence rule. But that necessarily doesn't allow any new capabilities. Randy notes that declaring an object in an inner scope that has an outer scope's accessibility would be new. Ed notes that the entire idea is to provide a capability that we don't have now (testing assignability on generic private types), so it seems unlikely that it could be completely equivalent to existing constructs.

We discuss whether sliding should or should not be involved in this case. It seems to depend on the use, which is hard to define. Steve notes that if one imagines that this should be equivalent to putting a handler around an assignment and doing the alternative only if an exception is raised, then sliding should be allowed.

Steve suggests that he would be more comfortable if this attribute was only allowed in a membership test (or possibly other restricted contexts). Randy notes that was one of the ideas explored in alternative 1.

Allowing the declaration of components or parameters with an attribute like this seem problematical (and would allow writing things not otherwise allowed).

The discussion is inconclusive, and Tucker will need to take this back to consider more of the corner cases.

Keep alive both alternatives: 10-0-0.

AI22-0077-1/01 Discussion on Accessibility Checking

We discuss how to handle this. No Action seems wrong as we do want to handle these topics, but not here. Hold seems wrong since those are expected to come up again someday; we review those periodically to return them to active status.

This is really an organizing paper that shouldn't have been an AI in the first place. We think such things belong in the Google Drive for the appropriate language study group.

Tucker adds a note to the AI saying that this is not appropriate as an AI. Now No Action feels better.

No Action: 9-0-2. Jeff and John abstains.

Justin says that his sister has gone into labor, so he has to leave.

AI22-0082-1/02 Problems with nonlimited reference types

Steve updated the AI with the list of changed paragraphs.

Randy notes that he researched why we had not made these limited in the first place. He was not able to find anything written that explains that. He believes that the concern was centered on the build-in-place rules that apply to limited functions, but as far as he can tell that was more FUD than a specific problem.

Steve notes he ran the AdaCore test suite with these changes. Only two minor problems showed, both of which would be easily fixed.

Gary notes that the !discussion says "If the compatibility impact…" That's talking about the solution not taken. That should be clearer. Tucker leaves a comment in the AI telling the editor to improve the offending paragraph.

Approve AI with changes: 11-0-1. Gary abstains.

AI22-0083-1/03 Treat dynamically-tagged expressions as class-wide in various contexts

A function call with a controlling result can be dynamically tagged if it has a class-wide actual parameter; it is odd that these aren't allowed in various contexts where class-wide things are needed.

Steve asks if there is a compatibility issue. Yes, it is possible for a previously legal call to become ambiguous. For example, if a package P declared a Union that returned class-wide, and Union as defined in these examples with a controlling result is in package Q, and the Union that returns a controlling result, a context that calls Union and has use clauses for both P and Q would now be ambiguous whereas it would have worked previously. This seems highly unlikely, but should be mentioned in the !discussion.

Niklas notes that the lead-in 4.6(21/3) talks about "both being class-wide", which also needs to be changed so that the bullet 4.6(23.1/2) is effective. The obvious rewording is awkward, and we no longer wordsmith on the fly, so this AI needs to go back to Tucker.

Approve intent: 11-0-0.

AI22-0085-1/02 Various presentation issues

Randy explains each of these.

Randy wonders whether Wide_Wide_Character'Last is well defined with this change. Tucker thinks that the number here doesn't define 'Last anyway.

Tucker and Jeff both suggest that "correspond to" should be changed to "include". Tucker suggests adding "Wide_Wide_Character has 2**32 distinct values."

Steve would like the same sort of wording added to other character types. He worries about machines that have extra bits. Randy notes that on the old Unisys machine, Character was a 9-bit type with 8-bit values. Tucker mentions the 24-bit machine used on the Patriot missiles. Could Wide_Wide_Character be implemented with a 24-bit type? That's clearly not allowed with the definition we just added, but whether it could be implemented with a 36-bit type is less obvious.

Steve would like an answer as to whether the 'Last values for Character and Wide_Character are the same on all machines, or based on the underlying representation.

Randy notes this is getting away from presentation. Tucker notes that it is associated with the change for Wide_Wide_Character, but not if we change other types. We don't change other types, Steve is told to submit a proposal to cover the other types if he is sufficiently concerned about obsolete machines.

Turning to the rounding wording, Steve suggests moving the System.Priority'Last earlier in the sentence. That eliminates the parsing problem.

Approve AI with changes: 11-0-0.

AI22-0086-1/02 Conversions of general access-to-object values

Steve had originally raised the issue. Randy explains the problem, an unconstrained type could have a Dynamic_Predicate.

Tucker wonders if this should be legal. He worries that the idea that "unconstrained" makes it a superset is no longer true because of the presence of predicates. He notes that SPARK tries to enforce Dynamic_Predicates always. So we need the predicates to "match" somehow.

Randy had noted that Object_Size needs to match, lest there is a massive problem. This was the primary reason that Object_Size participates in static matching; we cannot have an end-run around that rule. Randy says that we don't require compilers to support different object sizes for discriminated types, but since we allow it, we have to have the semantics well defined in case a compiler wants to support such a thing.

Specifically, the designated objects had better have the same in-memory size and representation, otherwise one could have an assignment that writes a larger object into a smaller one, overwriting memory that doesn't belong to the object.

Steve will take the AI.

Approve Intent: 11-0-0.

AI22-0087-1/01 Text buffers are initially empty

Randy was working on a presentation issue, and noted that there is no initial state defined for a text buffer. Tucker notes that when we simplified these packages, we removed some of the subprograms, and the contracts that used those subprograms were also eliminated, without being replaced.

We'd rather use English instead of contracts at this point, so we don't change these packages unnecessarily.

Tucker would like to add some wording to the summary explaining that junk wording was also removed. He does that.

Approve AI with changes: 10-0-1. Justin abstains.

AI22-0088-1/01 Privacy and Put_Image

Tucker notes that this came out of a query from an AdaCore engineer. He thought that it made it valuable to clarify it if it is unclear to knowledgeable people.

Steve asks that the wording change be marked as redundant.

Approve AI with changes: 11-0-0.

AI22-0089-1/02 Current Instance for singleton task/protected objects

Tucker explains that there are several rules that depend on the current instance of single tasks and single protected objects being defined. They could be changed by talking about single tasks explicitly, but that's wordy and not really helpful to the reader. So we define the term in that case.

Niklas notes that the return expression in the example is wrong; Tucker fixes it.

The syntax is "single_task_declaration" rather than "singleton_task_declaration". We remove the "ton" from some of the wording and the subject.

Approve AI with changes: 11-0-0

AI22-0090-1/02 Allow No_Return aspect on formal subprograms

Someone asks about pragma No_Return. Randy looks and sees that it takes a "*subprogram*_local_name". To allow it to apply to a type would require rewriting J.15.2. Tucker suggests that we don't bother with that.

Randy notes that adding capabilities to obsolete things seems weird. So the consensus is to not make a change to pragma No_Return's definition; it can be used on generic formal subprograms (which are subprograms), but not on access-to-subprogram types.

Jeff notes a capitalization problem in the example.

Niklas notes that there is an extra "is procedure" in the example.

Approve AI with changes: 10-0-0.