Minutes of ARG Meeting 63

Lisbon, Portugal

11-13 June 2023

Attendees:

In-person: Steve Baird,, John Barnes (Block J), Randy Brukardt, Jean-Pierre Rosen (All but Block J), Tucker Taft, Tullio Vardanega

Remote: Raphael Amiard (Block E after 10:30, G, H), John Barnes (Block B, D), Arnaud Charlet (Block G after 14:20 until 15:30), Jeff Cousins (Block E after 9:25, F, G, H), Gary Dismukes (Block C, D, G, H), Claire Dross (Block E, F, G, H), Ed Fish (after 16:15 in block D, G, H), Brad Moore (All blocks), Richard Wai (Block C, D, G, H until 16:28).

Observers:

In-person: Alejandro Mosteo (All but Block J), Pat Rogers (Block J after 11:04).

Remote: Niklas Holsti, Robin Leroy (Unicode Consortium Liaison to SC 22 – Block F), Erhard Ploederder (Block G, H).

Meeting Summary

The meeting convened on Sunday, 11 June 2023 at 9:12 hours WEST (Western European Summer Time) and adjourned at 12:18 hours WEST. The meeting was held in a conference room at the Fenix Lisboa Hotel in Lisbon, Portugal; remote attendees used Zoom. The meeting covered the entire primary agenda, along with some of the additional topics from the ARG Github Issues.

The meeting was organized in blocks for the convenience of the remote attendees. The actual blocks used (whose times varied slightly from those in the final agenda) were:

Block A	June 11, 09:12 to 10:45
Block B	June 11, 11:00 to 12:15
Block C	June 11, 13:45 to 15:25
Block D	June 11, 15:43 to 17:20
Block E	June 12, 09:06 to 10:46
Block F	June 12, 11:00 to 12:21
Block G	June 12, 13:54 to 15:30
Block H	June 12, 15:45 to 17:05
Block I	Canceled
Block J	June 13, 10:40 to 12:18

AI Summary

The following AIs were approved:

AI22-0015-1/01 Innermost master of the call is ambiguous (7-0-1)
AI22-0051-1/04 Preelaborable_Initialization and contract aspects (6-0-0)
AI22-0053-1/04 An unintended consequence of AI12-0101-1 (6-0-0)
AI22-0056-1/04 Automatic creation of constructor functions (6-0-0)
AI22-0062-1/02 Clarify “ceases to exist” definition (6-0-0)
AI22-0064-1/03 Basic rules for parenthesized expressions (6-0-0)
AI22-0065-1/02 Specialized Needs Annexes should be normative (10-0-0)
AI22-0074-1/01 Postcondition error in Ada.Containers.Hashed_Sets (6-0-0)
AI22-0078-1/01 Decimal conversions for Big_Reals (6-0-1)

The following AIs were approved with editorial changes:

AI22-0025-1/02 Accessibility of generalized iterators (8-0-0)
AI22-0033-1/02 Additional terms and definitions (9-0-0)
AI22-0034-2/02 Implementation model of dynamic accessibility checking (8-0-0)
AI22-0045-1/05 Issues with pragma placement (7-0-1)
AI22-0049-1/02 Seconds function with Time_Zone parameter (7-0-0)
AI22-0057-1/02 Floor and other rounding attributes for fixed point types (9-0-0)
AI22-0058-2/01 Wording for checking Task_Ids (8-0-0)
AI22-0067-1/01 The nominal subtype of an aggregate (6-0-0)
AI22-0069-1/02 Empty subsequences in parallel reduction expressions (6-0-0)
AI22-0073-1/01 Referencing the Unicode Standard (8-0-0)
AI22-0079-1/01 Parameters of a protected type (7-0-0)

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

AI22-0055-1/03 Usage Advice (9-0-1)
AI22-0059-1/01 Parallel_Calls aspects for types (8-0-1)
AI22-0076-1/01 Restricting dynamic accessibility checks (8-0-0)

The following AIs were discussed and assigned to an editor:

AI22-0022-1/02 Difficult example issues from WG 9 review
AI22-0026-1/01 Problem with nested type extension check
AI22-0063-1/02 Font alone should not differentiate terms
AI22-0071-1/01 Testing assignability and returnability
AI22-0072-1/02 Meaning of direct references to components
AI22-0075-1/01 Explicitly aliased results
AI22-0077-1/01 Discussion on Accessibility Checking

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

#1 – No_Return on access-to-subprogram type?
#5 – Class-user defined literals
#8 – Allow downward conversions of dynamically-tagged expression
#13 – Named parameters in reduction expressions
#42 – Potentially confusing definition of "declarative region"
#45 – Name resolution vs. legality for aspects, attributes, and pragmas
#53 – Allow prefixed views for more types
#61 – Class-wide container aggregates

The following Github Issues were discussed without a resolution:

#37 – New Unicode-related String Interpolation standard – consider for Ada?
#40 – Some kind of Univ_String to provide better approach to support full Unicode?

The following AIs were discussed and voted No Action:

AI22-0058-1/03 Preconditions for checking Task_Ids (8-0-0)

The intention of the following AIs were voted, but then were discussed again later in the meeting (the final results are above):

AI22-0056-1/03 Automatic creation of constructor functions (7-0-0)

The following AIs were discussed and assigned to an editor, but then were discussed again later in the meeting (the final results are above):

AI22-0056-1/02 Automatic creation of constructor functions

Detailed Minutes

Welcome

Steve welcomes everyone to this meeting.

Apologies

Jeff has various conflicts; he is on vacation on Sunday and has babysitting on Tuesday (he plans to attend Monday). Various other people will not attend Sunday sessions.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 6-0-0.

Date and Venue of the Next Meeting

Our next electronic meeting is proposed for Thursday, October 5 with the usual time (10:30-13:30 EDT [-4 UTC]) and method (Zoom). There are no objections to this date.

Standard Report

Randy gives a short report. 8652:2023 was finally published in May. He still has to created final versions of the RM and related documents (the introduction needs to be updated, and a few identified typos fixed). He also has been contracted to produce a reformatted version for Springer publication. He expects these to be finished in July.

User Community Input Working Group Report

Richard is not here, so Tucker gives a few words on the UCIWG. He notes that the documents describing the processes have been updated. The process is in use for all ARG work at this point.

Future Directions for the ARG

This session took up all of Block G and most of Block H.

The following topics were discussed:

AdaCore Perspective

Raphael Amiard gives the AdaCore position. How does the ARG want to deal with the possibility that there are no full Ada compilers? The ARG could continue as it has, but AdaCore could potentially reduce support. And no one would use the full Standard. The ARG would presumably pursue support elsewhere.

Raphael says that alternatively we could adopt a much slower evolution more like C. That is, more maintenance than new features. New features would be small and incremental. AdaCore might do big changes internally, but it would not do those through the ARG.

Richard says that implementers are mostly just not implementing something. Whereas AdaCore seems to want to do different things from the current directions. He wonders if it would be better for AdaCore to fork in that case (and call it something else).

Raphael says that is a possibility but that given that there aren't any other full implementations it doesn't seem that it would help. At this point, AdaCore is only working on small extensions that wouldn't be different enough to consider it a fork.

Tucker says that AdaCore (product management) is not wanting to make an investment in large new Ada features. Smaller features might be submitted to the ARG.

Tucker notes that AdaCore is already working on the restrictions against dynamic accessibility checks.

Tucker wonders if we want to find a group to implement the full Ada 2022, or possibly adopt a different technology. What would AdaCore's position would be if an independent group was to update the GNAT front-end to support the full Ada 2022 language?

Claire wonders if people would rather use a newer language. Claire doesn't think that people would use such a compiler. Richard thinks that much of the issue is awareness.

Raphael thinks that one can’t compete with a language like Rust, simply because of the resources that they have. The whole user experience, not just the compiler and features, matters.

Raphael notes that there is a lot of value to keeping the Ada heritage alive, but there doesn’t seem to be enough demand for new features.

Tucker notes that there is a large Ada community as a whole, not just AdaCore. The ARG is serving the entire Ada community; there doesn’t necessarily need to be a business case for it. There are a lot of small language communities.

Raphael says that such communities don’t bother with formal definitions of things. Tucker notes that the quality of the reference manual is an important part of Ada. Richard agrees with Tucker.

Jean-Pierre notes that popularity of languages tend to be measured on lines on Github. Industrial users aren’t posting on Github. Plus many things (like Javascript) is mostly copied. Raphael says that they are not selling to any new Ada users. Most of the customers seem to be trying to move to Rust.

Raphael notes that a lot of compilers are free these days, so there isn’t a lot of money to be made there anyway.

Richard notes that we don’t want to be adding features for the purpose of keeping up with other languages. Just adding a feature doesn’t have much effect on languages uses. We don’t want to just pile in all of the most popular features.

Richard notes that Rust has a loose organization that makes keeping a focus hard.

Arnaud doesn’t think that popularity is the right issue. He thinks that Ada is not providing the right building blocks for some projects. He talks about Wide_Wide_Whatever as an example of annoyances using Ada.

Ed Fish notes that it would possible to make a meta language to describe a lot of functionality for Ada.

Arnaud suggests that it is getting too complex to add new things into Ada because of the many interactions between existing features.

Richard notes that there are two extreme views. One is that Ada has very strict software engineering views, and one is that we should own that and live with that wherever it goes. The other possibility is to start over without worrying about compatibility, and essentially remove a lot of stuff.

Arnaud says that we need to find ways to simplify the Ada language, whether that is through the ARG or some other means.

Raphael doesn’t think a new compiler is possible. He wants to decide how we want to work on new features.

Randy notes that we haven’t worked on any significant new features to date; we’ve been waiting for the publication of Ada 2022 before doing anything significant. So it is premature to complain about what the ARG IS doing, we haven’t decided anything at this point.

Arnaud says that this is the point when we need to decide how we are going forward.

Tucker reraises the question of another implementation. Arnaud notes it could make sense to translate “new Ada” into “old Ada”. Raphael thinks it could be helpful to allow more people to experiment with new features in GNAT.

Ed Fish notes that Diana was an interesting technology. Steve notes that Libadalang provides similar functionality.

Tucker wants to know whether a group was working on such an enhancement to GNAT code, would AdaCore support this activity, or at least not interfere with such an activity. Arnaud says that AdaCore would provide some help this way. Raphael notes that some ways to help people do that would be of interest. It is important to prototype possible new features.

Tullio tries to summarize the discussion. The ARG exists as the technical arm of WG 9. AdaCore has decided that it was invested too much into the ARG at this point. The technical work needs an implementation, otherwise it is just an intellectual exercise.

Tullio continues, noting the tension between the AdaCore positions and the ARG position. One option would be for AdaCore to forget the standard and split off and do its own thing. Another option is to specify a set of profiles.

Tullio now puts on his Ada-Europe hat. If Ada-Europe can help getting a sense of what Ada users need, it would want to do that.

Tullio asks if AdaCore would want to split off to do its own (non-ISO) thing.

Arnaud says AdaCore would like to keep the Ada Standard, it helps in various ways.

Arnaud notes that Ada community includes all AdaCore customers. And this is a very important part of the community. He also notes that AdaCore writes a lot of Ada code every day.

Erhard has been working with other languages. If AdaCore doesn't want to implement something, that is no problem. If AdaCore were to change anything, then things are worse. In Rust and Python, you don't have that stability -- the implementations can do something different than the definition. Tools writers depend upon the stable definition of the language. He notes that there are a lot of tools writers that are part of the Ada community. He appeals to AdaCore not to change the language.

Arnaud claims that Ada is not a good language for writing tools. They need to add things.

Erhard says that such things need to be sent through the standardization process.

Arnaud only sees two options for use as a foundation for new front-end technologies: GNAT & Libadalang

Arnaud also observed that Ada has lots of aspects, attributes, pragmas, etc. to provide functionality that other languages might provide via libraries.

Richard thinks he is hearing that we need to put more of emphasis on prototyping of features before standardizing.

Erhard notes that there is a risk that tools don't support vendor-specific extensions. It's better to have them standardized.

Raphael again says that it is important to slow down the process of making changes (more than for Ada 2022).

Tucker notes that in the past we had the sense that what we were doing was what AdaCore wanted as well as what the community wanted. At some point that changed and we didn't notice that quickly enough. He notes that we need to rectify that. And we also do not want to rely on a single implementation for prototyping.

Succession

Randy notes that JTC1 is considering adopting term limits for **all** JTC1 positions. (Currently, only certain positions, like convenors, are term-limited). While we generally consider the ARG as organized outside of JTC1 (since it updates the RM, which is outside of JTC1), that does not apply to true JTC1 positions like Project Editor.

Thus, Randy may not be able to continue as Project Editor in the future. He also notes that he doesn't plan to work full time after he turns 70 (5 years in the future). So we need to look at approaches to find people to do some (and eventually all) of these jobs that he currently is doing.

Richard notes that it could be useful to carve out some smaller jobs and get people to help with them.

Tucker notes that we can try to find younger people to help in these roles. (Both Editor stuff, and implementation groups.) Ada-Europe could help publicizing these roles.

Tullio notes that it would be useful to create vacancies for these roles, and then try to find help.

Raphael notes that maintaining the RM is another task. Randy is almost the only one who knows how to do that maintenance. [Randy notes that the source of the tools is all written in Ada with an open-source license; several people have used those tools to make other versions of the RM.]

Raphael says that AdaCore might be able to help with some of the ARG infrastructure (notably the online RM - generation and hosting).

Future meetings

Should we continuing to have face-to-face meetings? The attendance at this one is not that good. Steve notes that all-day meetings mean that someone attending remotely finds it difficult to attend the entire meeting. [Brad Moore has heroically been attending starting at 02:00 his local time.]

Tullio notes that for a good group, we should continue to have some face-to-face meetings.

Tucker suggests possibly having two meeting places partitioning a meeting that way, with five hour meeting so everyone could be attending.

We certainly want such meetings, at least once every two years. Probably at Ada-Europe. Possibly with a gathering in the US as well.

Side-issue, Raphael would like to have some action items from the Future Development. We decide to make more concrete proposals at our next meeting (October). He would like to have some smaller changes to make work more useful.

New feature development

Brad wonders what the development effort would be on implementing the unimplemented Ada features. Tucker says for parallel, it isn't a lot, for other things, it's hard to say. He thinks that Bob Duff could help with GNAT, at least at the simple levels.

Tucker will take an action item to look for people who want to work on GNAT and/or new technologies, assuming that some resources can be found.

A tangent on Graal ensues. It is one of several technology packages that could be used.

Randy asks if we should be starting formal development of significant new features? Tucker and Tullio rephrase the question as should we wait until an implementation of Ada 2022 is implemented. We think that implementations need to catch up to the state of the standard before it makes sense to do anything serious with new features.

So that answer is essentially no, we're not going to do a lot of new feature development for now.

On-line only AI processing

The proposal is to process easy AIs outside of meetings.

We discuss this for a while. We don't think that there is enough advantage to doing AIs asynchronously. They don't take that long when there are not many errors. And if there are a lot of errors, that does not reflect well on the proposal; careful review is warranted in that case.

We should encourage typo corrections to be made as suggestions on the Google Docs. This should simplify meetings as we would only need to talk about those that are not obvious fixes. Gary suggests that we stop reworking text more quickly. The suggestion is that anyone can break in to suggest that we take it off-line when rewording starts to dominate a meeting.

These two suggestions should help move the meetings faster. But it is valuable to encourage everyone to read the AIs and think about them. And having a deadline of a meeting helps that happen.

We think (as a group) that it is better to take the actual votes at a meeting with most of the group present it gives everyone a chance to weigh in.

Appreciations

Thanks to Ada-Europe for the fine facilities and refreshments.

Thanks to Steve for his tireless work as Rapporteur.

Thanks to Randy for his even more tireless work as Project Editor, ARG Editor, and so on.

Thanks ISO for finally publishing the Ada 2022 Standard (in 2023).

Thanks to all of the in-person and remote attendees, especially Brad who managed to attend every minute, even in the middle of the night (for him).

Unfinished Action Items

Many of the previously long-term unfinished action items were finished for this meeting. There still are a few unfinished action items. Tucker notes that he is close to finishing the OpenMP Technical Report. He notes that it is somewhat of a moving target, it needs updates each time OpenMP is updated.

Two people still owe ACATS Tests: Ed Schonberg and Justin Squirek. We decide to retain these on the action item list.

Current Action Items

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

All:

- Propose simplifications to our procedures to make our meetings more efficient.

Steve Baird:

- AI22-0026-1
- Craft a To Be Honest for AI22-0067-1
- AI22-0076-1 (with assistance from Tucker Taft)
- Github Issue #13
- Github Issue #53 (assist Gary Dismukes)

Randy Brukardt:

- AI22-0072-1 (with help from Tucker Taft)
- Github Issue #45

Editorial changes only:

- AI22-0025-1
- AI22-0033-1
- AI22-0034-2
- AI22-0045-1
- AI22-0049-1
- AI22-0057-1
- AI22-0058-2
- AI22-0067-1
- AI22-0069-1
- AI22-0073-1
- AI22-0079-1

Gary Dismukes:

- Github Issue #53 (with help from Steve Baird)

Edward Fish:

- AI22-0022-1

Alejandro Mosteo:

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

Ed Schonberg:

- ACATS C-Test(s) for filters

Justin Squirek:

- ACATS C-Test(s) for Wide versions of Command_Line, etc. (or choose another, untested area)

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.
- AI22-0055-1
- AI22-0059-1
- AI22-0063-1 (Split work with Richard Wai)
- AI22-0071-1
- AI22-0072-1 (assist Randy Brukardt on request)
- AI22-0075-1
- AI22-0076-1 (assist Steve Baird)
- AI22-0077-1
- Github Issue #5 (assist Alejandro Mosteo as needed)
- Github Issue #8
- Github Issue #42
- 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 2022 AARM, the number refers to the text in draft 35 of the Ada 2022 AARM (the submission draft). 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 2022 AIs

AI22-0015-1/01 Innermost master of the call is ambiguous

Tucker noted that some of the wording was confusingly close to a new official term.

Approve AI: 7-0-1 Gary abstains.

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

Ed Fish explains his compilable examples.

We need to know where this goes in the RM (and which parts go in the RM).

Ed will take this AI back and make the obvious changes.

[Editor's note: We generally avoid uses of things not yet defined in the RM in examples. So, for instance, Ada.Task_Identification should not be used in examples for Section 9 (since it is defined long after section 9).]

Keep alive: 9-0-0.

AI22-0025-1/02 Accessibility of generalized iterators

The qualifier of "omission" is incorrect and should be removed.

Approve AI with changes: 8-0-0.

AI22-0026-1/01 Problem with nested type extension check

Typo (Tucker): in Issue: ... object of a tag {that} is shorter-lived...

Randy doesn't understand. The accessibility is passed in from the type conversion in this case. So the check has to be on the allocator.

Steve wonders if the wording should be access-to-classwide. But that doesn't make sense, the example is access-to-specific, and it clearly causes the problem. Steve eventually agrees.

Steve notes that this example could be caught statically, since only null would not raise an exception. He notes that a runtime check would still be needed (one doesn't always know what someone tries to return).

We look at this idea, but it seems to be a different issue. It might make sense to make such a check, but it seems that is it not necessary or sufficient to fix the problem here. Steve is encouraged to look at this conversion to see if a compile-time rule could be crafted (reducing the need for dynamic checks is always a good thing).

[We didn't take a vote on the original AI, waiting for Steve's update/take.]

AI22-0033-1/02 Additional terms and definitions

Tucker does not like the first definition. He wants to allocate them from a storage pool.

> Expression that allocates and initializes an object and yields an access value that designates it.

Next: We wonder if "denote" is the actual term. It is, declared in 8.6(16). There are other terms, but they are worse.

"denote an entity" would be easier to define.

Tucker would like "denotes" and "designates".

He defines those.

"overloaded subprogram" isn't used in the Standard, and these terms need to be used exactly in the Standard.. We end up using "overloaded", and then add "declarative region".

We also add "operative constituent".

Approve AI with changes: 9-0-0.

AI22-0034-2/02 Implementation model of dynamic accessibility checking

The goal here is to describe a model that is a relatively straightforward implementation. Hopefully, implementers can find more efficient implementations.

In this model, accessibility is specified by level-object. The object points to the parent object – including across tasks. A thread-local storage object always points at the most recent level-object.

For tag-accessibility, a level-object is only needed when it is more nested than any of its ancestors.

Approve AI with changes: 8-0-0.

[We should have voted AI22-0034-1 No Action at this point, but we didn't – Editor.]

AI22-0045-1/05 Issues with pragma placement

The WG 9 review identified some additional issues with this AI, so it is back to us for additional clarifications.

The most important comment was the following: Control pragmas say they are allowed anywhere that other pragmas are allowed; declare expressions say that only executable pragmas are allowed. One of these must be wrong.

Randy was unsure whether we wanted to allow control pragmas in declare expressions. Some of these (like List and Page) are lexical in nature and there doesn't seem to be any reason to restrict them. On the other hand, we were trying to minimize the work needed for implementations for declare expressions.

Tucker proposes changing control pragmas to be a specific list of locations. That would eliminate the conflict.

Niklas wanted to use statement rather simple_statement. Is statement a syntax non-terminal? Yes. But we discover the labels are in that production. We need to allow labels on pragmas – it's allowed in Ada 2012 and 2022, and it be incompatible to change that. So we really need it to be simple_statement, but the reason for that is completely wrong. Tucker updates the AI accordingly.

There also was a comment on 13.1(4). We decide that the wording is OK, this is redundant. It is not complete, but it is correct.

Approve AI with (many) changes: 7-0-1. Brad abstains.

AI22-0049-1/02 Seconds function with Time_Zone parameter

In !Issue, there are double double quotes for some reason, use only a single set.

Should this be a BI or Amendment? John says it matters if we want it in his book. Randy thinks a BI is weird one, as this package has been around since 2005. Tucker proposes that we make it a BI, and include a testing statement that we do not suggest testing for Ada versions earlier that Ada 2022.

So we do that. Steve also asks to explicitly say that it is optional for older versions.

Approve AI with changes: 7-0-0.

AI22-0051-1/04 Preelaborable_Initialization and contract aspects

Tucker explains the problem and the proposed solution.

The permissions allow you to defer or omit the checks on P_I objects.

Steve notes that this will impact implementations, as they have to treat P_I objects differently than other objects. We decide this is necessary to preserve the guarantees of Preelaboration.

Approve AI: 6-0-0.

AI22-0053-1/04 An unintended consequence of AI12-0101-1

Tucker and Steve explain the problem. We need a composed equality to do the same thing as the equality for a type. But that is not true for a hidden equality of an untagged type.

We use the narrowest possible rules to minimize compatibility problems.

Approve AI: 6-0-0.

AI22-0055-1/03 Usage Advice

We need to adjust the wording to meet the directives.

Tucker proposes a Usage section containing the appropriate notes and many of the examples.

Gary asks if we are sure that this change would meet the JTC1 requirements. Randy says that the ISO editors told us several times that we could move the notes to the normative part, we just didn't want to do that at such a late point. So he is confident that would be OK. The examples changes that Tucker is suggesting are not so clear.

Approve intent: 9-0-1 (Gary abstains).

AI22-0056-1/02 Automatic creation of constructor functions

Randy explains the problem and the proposed solution. The idea is to extend the automatic creation of functions for derived types to be possible for all types. This is especially important for mix-in generics, where overriding the functions isn't possible.

The name of the aspect is too long. Randy agrees. Please suggest a better name, he couldn't think of one. Tucker thinks the name needs to talk about constructors, since that is what it is about. Construct constructors is that we're doing. Constructor_Extension is suggested.

Constructor is not a term. Perhaps it should be.

The AI is given to Tucker.

Keep alive: 7-0-0.

AI22-0056-1/03 Automatic creation of constructor functions

[Editor's note: This was discussed as "AI22-0056-1b" so that the intermediate versions could easily be preserved.]

On Monday morning, Tucker explains his changes to the AI. He defined the term “constructor function” and used that in various wording. He also replaced “default extension initialization” with “automatic constructor extension“. And renamed the aspect.

If the partial view doesn’t have it, the full view could have it. That happens currently for null extensions, and compatibility concerns means that has to stay. Tucker wonders if that is more of a bug than a feature. We decide that partial view and completion should match (except for null extensions).

Jean-Pierre notices that others => <> would work for a null extension, so they don’t need a special mention.

Brad wonders if this would be good as a configuration pragma, since this seems to be the way most code would want to be. Randy didn’t initially propose that because it would be complication. Tucker notes having this be the default makes the default case less safe (more uninitialized components can happen), so this does not seem to be a good idea for Ada.

Approve intent: 7-0-0. (Jeff doesn’t vote, he just arrived).

AI22-0056-1/04 Automatic creation of constructor functions

[Editor’s note: This was discussed as “AI22-0056-1c” so that the intermediate versions could easily be preserved.]

On Tuesday, Tucker explains the revised AI.

If the function is nonabstract, then the nonabstract constructor is constructed (whether any of the types are abstract does not matter).

Steve notes that this aspect should not be allowed for an extension that has new discriminants, because the others => <> is illegal in that case. He notes that the term null extension excludes records that have new discriminants. We need to exclude that.

We need to make this apply to the full type when the partial view doesn’t have discriminants. It cannot have discriminants in that case. Tucker adds a legality rule to that effect.

We think that the model Tucker described that all nonabstract routines are inherited that way is indeed what happens for null extensions.

Approve AI with changes: 6-0-0.

AI22-0057-1/02 Floor and other rounding attributes for fixed point types

We discuss the AI. These attributes are confusing and possibly wrong if integers are not model numbers of the fixed point type. We could come up with a definition, but it wouldn’t be very useful, and implementation would be problematic (one would need accuracy rules for what should be an exact conversion).

The dynamic check for generics needed rewording (see the Google Doc).

Approve AI with changes: 9-0-0.

AI22-0058-1/03 Preconditions for checking Task_Ids

A precondition checks for termination too early (the task could terminate between the precondition check and the access to retrieve the information). Thus, this is replaced by AI22-0058-2.

No Action: 8-0-0.

AI22-0058-2/01 Wording for checking Task_Ids

Randy notes that the original AI22-0058-1 introduces a race condition, so we can't use a precondition to check termination. It's not worth changing to a mix of preconditions and English. So this AI just adjusts AARM notes.

Approve AI with changes: 8-0-0.

AI22-0059-1/01 Parallel_Calls aspects for types

Tucker explains the intent.

Steve wonders how this can be checked if the Storage_Pool is in a separate package.

After much discussion, we determine that it is ensuring that Parallel_Calls is true for storage pool routines, as opposed to applying it to them. The wording needs to make this clear.

"indicates" is weird, we try a few words and then settle on "requires" Parallel_Calls be True.

Tucker is unsure if Parallel_Calls is the right thing for storage pools; we need them to work on the same object called at the same time. We decide to send this back, Tucker will look at this again to figure that out.

Approve intent: 8-0-1 (John abstained).

AI22-0062-1/02 Clarify "ceases to exist" definition

We discuss the problem. Discriminant dependent components can be reallocated (or moved) on an assignment (and some implementations do that). We need to make execution erroneous in that case, in particular so that static analysis definitely knows not to depend on it.

Approve AI: 6-0-0.

AI22-0063-1/02 Font alone should not differentiate terms

Tucker explains that we are working on a list of words that are confusing. He appeals to anyone that finds any such terms, to send them to him and/or Richard.

Keep alive: 10-0-0.

AI22-0064-1/03 Basic rules for parenthesized expressions

Randy explains the purpose is to give the rules for parenthesized expressions. We give rules for all of the specific properties, it's weird to not give the general case.

This AI was split last time, this part has not changed much.

Approve AI: 6-0-0.

AI22-0065-1/02 Specialized Needs Annexes should be normative

Tucker explains that he is proposing that we eliminate any talk about optionality in the RM. The conformance of each thing is defined separately. He notes that implementations are choosing individually what to implement; the all-or-nothing focus of Ada in the past does not reflect reality.

Approve AI: 10-0-0.

AI22-0067-1/01 The nominal subtype of an aggregate

This matters now because of rules in Unchecked_Union, and eventually for case statements on record types (which is proposed). Tucker explains the proposal.

He explains that he had to use different rules for each type of aggregate.

Steve notes that the bounds of subaggregates have to agree at runtime, but they don’t have to be the same at compile-time.

This doesn’t matter for any particular usage. (An exception will be raised.) Steve says that the legality of a case statement could be affected. Randy doesn’t think these bounds are static (that would have to be added). We think there should be a To Be Honest to explain this case (since it can’t matter, the subtype can’t be used dynamically if an exception is raised during the evaluation of the aggregate). Steve will craft a To Be Honest note to explain this.

Tullio notes that there is a sentence that needs a verb, after the example. We added a bullet to the quoted text to make it more obvious that it is a fragment.

Approve AI with changes: 6-0-0.

AI22-0069-1/02 Empty subsequences in parallel reduction expressions

4.5.1 should be 4.5.10 in the wording. The third paragraph in the wording is missing the Redundant bracket.

Approve AI with changes: 6-0-0.

AI22-0071-1/01 Testing assignability and returnability

We discuss this for quite a while. There does not seem to a use case for ‘Returnable.

There is interest in pursuing ‘Subtype. But that seems to be more general than we need for this specific problem. We already had that idea for Ada 2012 and decided not to pursue it. See AI12-0123-1.

Tucker would like to try the attribute again. We’ll construct an alternative to AI22-0071-1 to propose ‘Subtype to solve the Github problem (and others), and make no effort to solve the others.

We won’t vote on this since we don’t have anything to vote on.

AI22-0072-1/02 Meaning of direct references to components

Jean-Pierre complains about “most contexts” in the Summary. We decide that it applies except in representation clauses. (It appears to apply to aggregates, but those are selector_names, not direct_names).

Steve claims that one can refer to entries directly in accept statements. That is a direct name, so this rule has to include that.

Tucker claims that “current instance” doesn’t exist for single protected/tasks.

Randy and Steve note that this is important, we can’t “assume” that we know how these work.

Clearly, we have to do something separate for the single cases. It’s not clear how best to do that. We send the AI back to Randy to figure that out (with help from Tucker).

Keep alive: 6-0-0.

AI22-0073-1/01 Referencing the Unicode Standard

Robin Leroy (Unicode Consortium Liaison to SC 22) starts with a presentation on their most recent work.

Unicode is trying to create rules to make it hard to write code that looks the same but has different meanings. The group is working on recommendations for programming languages and environments. These are hard to understand, so they are trying to make it easier to follow those and provide help interpreting them.

These are new/updated documents, with documents to be finalized in July and published in September.

The proposal today is to change directly to reference the Unicode Standard. Once the documents are published, there will be other proposals to bring Ada in line with the recommendations, while preserving compatibility.

Turning to this AI. C and C++ have normative references to Unicode. While it may have been a problem to reference Unicode in 2005, it does not appear to be so today. Therefore, we should change to directly reference Unicode documents when appropriate.

Tucker asks whether we need to talk about 10646 at all. Robin notes that C++ had dated and undated references to 10646 and Unicode, so they had 4 character set standards in various places. Ada doesn't have that problem, and it is best.

Steve asks what C++ was doing. Robin says that they referenced UCS-2, which is now gone. So that had to reference an old standard in order to get that.

Robin has dated and undated references. Some things refer to specific chapters of the Unicode Standard, which differ between versions. Otherwise, Ada has things that allow using any version of 10646, those are handled by using undated references. [Editor's note: I don't believe Tucker's question was ever answered directly; if it was, I didn't record it. Nothing in the above seems to require the references to 10646, perhaps they could be removed?]

Eventually, we'll want to change to using XID_Start and XID_Continue, which are guaranteed to only grow (add new characters).

We joke about identifiers using emojis, including the Pile of Poo. C++ allowed that for a while (by not restricting the characters much), they now are restricting them.

Robin had to figure out where things are defined in Unicode. Simple case folding isn't properly defined in Unicode. He is working on getting that fixed. For now, we will use the place where it **should** be defined.

There aren't many locale-specific simple case folding maps; Unicode mainly just defines the one for Turkish (and not really that). So we should remove "locale-independent".

Approve AI with changes: 8-0-0.

AI22-0074-1/01 Postcondition error in Ada.Containers.Hashed_Sets

We discuss better ways to write this for a while, but eventually decide that changing the way this is written is risky and would require much additional change. (All similar containers would need to be changed the same way.)

Approve AI: 6-0-0.

AI22-0075-1/01 Explicitly aliased results

Tucker explains that this AI provides a way for functions to return writable variables.

It essentially allows returning a writable (or constant) object that will live at least as long as the point of the call (so it is safe to use).

"includes the reserved word aliased" is too vague (it could be construed to include things anywhere, including in the parameters.

Tucker thinks that we need to define “result profile”, and then use “result profile has aliased”. Randy wonders if that should be defined with extra syntax (since it would be very clear what it is), or define it semantically.

Steve asks about the level of a call. Randy asks what if someone tries to take ‘Access of it. These are the same. There needs to be a definition of the level of an explicitly aliased result. They are aliased, so we definitely need to know. Tucker adds a note that 3.10.2 needs a definition of the level of explicitly aliased results.

Steve wonders about nested extensions in a dispatching call (there the inherited function might have a different level, deeper level, which might get a different answer for the accessibility). Tucker adds that to his list of problems to check (he thought that there was no problem, but he can’t remember why he thought that).

Steve notes that one will have to look at generic formal subprograms with explicitly aliased results in order to make compile-time accessibility for calls.

Gary would like more worked out examples.

Interaction with build-in-place needs to be spelled out (there is no build-in-place).

Steve comments that the existing user-defined indexing is harder to use than necessary. We (the ARG) need to work harder to make user’s life easier (if there is a choice). This is such a case.

Keep alive: 8–0-0.

AI22-0076-1/01 Restricting dynamic accessibility checks

Tucker goes over the areas that need changes (restrictions) in order to eliminate dynamic checks for access types.

He then moves on to the restrictions themselves. For stand-alone objects, the intent is to use a static level. Randy wonders how that could be described as a restriction, it sounds more like a semantic change. Tucker says that is to be determined at some future point.

Randy thinks that this might be better handled as a profile. We already have a No_Coextension restriction, it hardly makes sense to define it a second time. A profile is a set of pragmas, so we could also have interesting other things that don’t look like restrictions.

Is this worth pursuing to write this up? Yes. Steve volunteers to take this AI. Tucker will assist.

Richard asks why we couldn’t use an allocator with access discriminants. Tucker notes that there is lot of finalization complexity associated with coextensions.

Approve intent: 8-0-0.

AI22-0077-1/01 Discussion on Accessibility Checking

This is an overview of accessibility checking.

Tucker goes over the various ideas associated with accessibility checking; some of which are included in other AIs on the agenda: AI22-0034-2, AI22-0075-1, and AI22-0076-1.

[We didn’t vote on this, it probably should get some sort of vote sometime. Else it will remain on the agenda and Tucker’s work list forever.]

AI22-0078-1/01 Decimal conversions for Big_Reals

The Big_Real to Decimal conversion generic is missing, this AI proposes to add it. We discuss for a long time why we have a “hypothetical floating point type”. After a long discussion, we decide that we wanted to treat this as a float to fixed conversion for accuracy purposes, and that is the reason for the hypothetical float type. So this is fine as it is.

Approve AI: 6-0-1 (Jean-Pierre abstains).

AI22-0079-1/01 Parameters of a protected type

Jean-Pierre says that he has examples where he declares a task object inside of the type itself. The Sieve of Eratosthenes is such a case. It seem safe to allow that as well. We definitely don't want to allow this in (most parts of) expressions, there is too much ambiguity between type conversions.

We decide to add subtype_mark of the subtype_indication of an object_declaration.

Tucker changes this wording into bullets. Randy notes that the JTC1 directives only allow a few ways to structure bulleted lists, and we often violate those. The question is whether all of the bullets should end with "or", or only the last. [Editor's note: The JTC1 directives actually don't say a lot about "lists", but the ISO House Rules have a lot to say. All bullets must end with either . (for full sentences), ; or , (for fragments). The penultimate item shouldn't end with "and" or "or" "except to avoid ambiguity"; they never mention ending **every** line with "and" or "or" as we often do. Cases where we have multiple sentences in bullets don't seem to be allowed at all. OTOH, the "House Rules" is an advisory invention, it can be worked around if necessary.]

Brad has some typos that are corrected in the AI.

Approve AI with changes: 7-0-0.

Detailed Review of Github Issues

#1 – No_Return on access-to-subprogram type?

Seems reasonable and not too hard. Tucker will take this AI. Make it a Binding Interpretation.

#5 – Class-user defined literals

Alejandro Mosteo already looked at this (according to a post in the issue). We assign this to him, Tucker will mentor him since he is new (will be made an ARG member at the WG 9 meeting that follows us).

#8 – Allow downward conversions of dynamically-tagged expression

Steve would like to identify the problem that this is solving (rather than the technical issue in question). Examples of the problem in practice would be very helpful. Tucker will take this AI.

#13 – Named parameters in reduction expressions

We don't want to generally allow named parameters in attributes, that would be a change that could be very expensive for implementations and no one has asked. But reduction expressions are new. And have explicit syntax.

We will not allow reordering (as with pragmas). They would be optional parts of the reduction expression name.

Tucker suggests "Reducer" and "Initial_Value", which is consistent with 4.5.10(23/5). We should also update most of the examples to use named parameters (it would be more readable to include Initial_Value => in them).

Make this a Binding Interpretation as Omission.

Steve will take the AI.

#37 – New Unicode-related String Interpolation standard – consider for Ada?

Robin notes that Unicode defines this to use for localized messaging. He thinks that it should be integrated with a localization framework. It is not directly relevant to Ada as it stands.

Tucker notes that there are other reasons where parameterizing strings makes sense. He was suggesting that we should consider using the Unicode standard as a starting point for interpolation for the Ada string. He notes pluralization as an example. Robin notes that doesn't work if you are doing localization.

#40 – Some kind of Univ_String to provide better approach to support full Unicode?

Robin has a comment in the issue about UTF-16. The surrogate characters have no use other that of extra characters, so UTF-16 and UCS-2 are compatible. For UTF-8 and UCS-1, the code points have different meanings depending on which is being used.

There are sets of code points (graphemes) that should stick together, even in UCS-4 (Wide_Wide_Characters). So one can't assume that plain indexing is safe.

#42 – Potentially confusing definition of "declarative region"

We discuss whether the proposed new wording would help.

This would be a presentation AI. Tucker is happy to be the stuckee on this one.

#45 – Name resolution vs. legality for aspects, attributes, and pragmas

We want to treat these as local_name unless otherwise specified. Add a blanket rule in 13.1.1, and check all of the aspects that are subprogram valued to see if they need to be otherwise specified. Randy notes we need to allow nested packages. Tucker says we should say within the immediate scope; that allows inside things, but not outside things. And other rules remain as Legality Rules.

Randy will take this AI. This is a Binding Interpretation.

#53 – Allow prefixed views for more types

Gary has written something for this in GNAT. Assign to Gary, with Steve as a backup.

#61 – Class-wide container aggregates

Alejandro Mosteo already looked at this (see issue #5). We assign this to him as well, Tucker will mentor him.

Randy notes that a reasonable approach here would be to treat such aggregates like tag-indeterminate functions. (That would be the case for class-wide user-defined literals, which are functions). That way, the specific tag could come from the LHS of an assignment or from another parameter in a complex call, maximizing the utility of such aggregates. (A specific tag is needed for the resulting object, of course.)