Minutes of Electronic ARG Meeting 58A

29 January 2018

Attendees: Raphaël Amiard, Steve Baird, Randy Brukardt, Jeff Cousins, Gary Dismukes, Bob Duff, Brad Moore, Erhard Ploedereder, Jean-Pierre Rosen, Ed Schonberg, Tucker Taft.

Observers: Peter Chapin, Justin Squirek.

Meeting Summary

The meeting convened on Monday, 29 January 2018 at 11:05 hours EST and adjourned at 13:30 hours EST. The meeting was held using WebEx. The meeting covered a small part of the overly ambitious agenda, but did cover all of the regular AIs.

AI Summary

The following AI was approved:

AI12-0247-1/01 Potentially blocking goes too far for Detect_Blocking (11-0-0)

The following AIs were approved with editorial changes:

AI12-0244-1/01 Check name for Value attribute (11-0-0)
AI12-0252-1/01 Duplicate interrupt handlers under Ravenscar (10-0-1)

The following AIs were discussed and assigned to an editor:

AI12-0020-1/01 'Image for all types
AI12-0079-1/05 Global-in/global-out annotations}
AI12-0208-1/00 Predefined bignum support
AI12-0242-1/02 Reduction expressions
AI12-0243-1/01 Subtypes as primitive arguments
AI12-0246-1/01 Fixed first indices for variable-length arrays
AI12-0248-1/01 Null array and empty container aggregates
AI12-0253-1/01 Saturation arithmetic for scalar types
AI12-0254-1/01 Bounded_Indefinite_Holders

The following AIs were discussed and voted No Action:

AI12-0245-1/01 “with and use” clause (9-0-2)
AI12-0255-1/01 Component iteration/reflection (8-0-3)

Detailed Minutes

Apologies

The management apologizes to the entire membership for this morning's Google Hangout fiasco. The original Hangout was limited to 10 attendees (a problem for a meeting with 13 people!), and the new hangout could not be heard by several people (apparently, so many participants overload the hangout software so that older computers cannot keep up). Luckily, Tucker Taft had set up a backup WebEx meeting.

Previous Meeting Minutes

No one has any changes to the minutes. Approve minutes: unanimous.

Date and Venue of the Next Meeting

Randy had arbitrarily picked February 26th as a suggested date for our next electronic meeting. Several people cannot make that date. We decide on March 5th, at an earlier time of 10:30 am EST as all of our west-coast members will be in Paris that day. (Randy grumbles about getting up in the middle of the night, that is, 8 am. Everyone laughs appropriately.)

The next in-person meeting will be in Lisbon, Portugal, June 22-24, following Ada-Europe.

ACATS Tests

Randy notes that it is time to start work on ACATS tests for Ada 2020. We had previously agreed to have the author create tests for all Amendment AIs. This has not been happening. What is our realistic plan?

Jeff: we should not put them in the Standard until there are tests, or at least lower the priority of AIs without tests. Randy (as editor) objects, as that would push most of the work of creating the Standard to the absolute last moment. Given that problems often show up in this process, it would be difficult to do that.

Tucker suggests that we tentatively approve AIs without insisting that they're in the Standard. Randy (again speaking as editor) points out that does nothing for his concern of doing the actual Standard at the last minute. Someone suggests mitigating that with version control, but Randy thinks that the AIs are too interrelated in general for that to be very effective.

Randy also notes that we have only about 16.5 months remaining to work on AIs. We're not going to be able to get all of the proposals to a finished state in that time.

We need to look at all of the proposed AIs as an entire group and try to set prioritization. We'll do that at the next meeting.

Randy will make a list of issues that need tests (already approved AIs) and circulate that for volunteers.

Unfinished Action Items

We skipped this usual agenda item for this meeting. Members are reminded as always to do their homework.

Current Action Items

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

Raphaël Amiard:

- AI12-0214-1 (split into two AIs; help from Tucker Taft)
- AI12-0218-1 (get Thomas Quinot to write a detailed proposal, hold otherwise)
- AI12-0253-1 (assist Ed Schonberg)
- AI to make type marks optional in object renames (see discussion of AI12-0236-1 of meeting #58)

Steve Baird:

- AI12-0016-1
- AI12-0020-1
- AI12-0208-1 (with help from Bob Duff)
- AI12-0210-1 (get guidance from the SPARK group, then update the AI)
- AI12-0243-1

Randy Brukardt:

- AI12-0017-1 (assist Florian Schanda)
- AI12-0112-1
- AI12-0254-1

- Circulate a list of Amendment AIs needing tests for Ada 2020 (for volunteers)

Editorial changes only:

- AI12-0244-1
- AI12-0252-1

Alan Burns:

- AI12-0230-1 (with assistance from Tucker Taft)

Bob Duff:

- AI12-0208-1 (assist Steve Baird)

Brad Moore:

- AI12-0234-1

Florian Schanda:

- AI12-0017-1 (with help from Randy Brukardt)
- AI12-0188-1 (see discussion of AI12-0189-1 in Vienna)
- AI12-0197-3 (lower priority than others)

Ed Schonberg:

- AI12-0253-1 (with help from Raphaël Amiard)

Tucker Taft:

- AI12-0020-1 (propose a stream-like interface as an alternative to a functional interface)
- AI12-0079-1
- AI12-0111-1
- AI12-0189-1
- AI12-0191-1
- AI12-0214-1 (assist Raphaël Amiard)
- AI12-0226-1
- AI12-0230-1 (assist Alan Burns)
- AI12-0235-1
- AI12-0246-1
- AI12-0248-1
- AI for 'Value for composite types, similar to AI12-0020-1 (see discussion of AI12-0020-1 in Unfinished Action Items of meeting #58)
- Potential AI to declare System.Storage_Pools.Subpools Pure (see discussion of AI12-0235-1 during meeting #58)

Detailed Review

The minutes cover detailed review of Ada 2012 AIs (AI12s). The AI12s 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 202x AARM, the number refers to the text in draft 12 of the Ada 202x AARM. 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 2012 AIs

AI12-0020-1/01 'Image for all types

Steve sent out e-mail discussing the model. Are there any questions/objections?

Gary comments that there has to be a runtime routine for each one. Steve thinks that streaming is the model, in general there would not be any code generated if it is not used. (At least, that's the way it works in GNAT for untagged types.) [Randy wishes that he had noted that while such routines would be generated for every type in Janus/Ada, bind-time dead code elimination would remove any that aren't called, even for tagged types.]

Erhard asked about how access value are specified. It just is a value of some sort (details still TBD). In particular, it does not try to walk through access values, as that could get stuck in cycles and the like. That's what Erhard was concerned about. Of course, a user could write an overriding 'Image to do a “deep” Image.

Steve suggests that these are “pure”, the compiler can assume the same value is produced on each call with the same object. Tucker wonders if these are functions? He suggests that a different interface might be worthwhile (an accumulator interface, adding pieces to string, much like streaming).

Bob thinks that this turns into I/O interface.

Tucker notes that this model would automatically compose like a stream. He is concerned that a pile of function calls returning strings would require lots of concatenations and dynamic storage allocation. It would be hard to write and inefficient.

Steve would like Tucker to sketch an example of this model.

AI12-0079-1/05 Global-in/global-out annotations

Tucker hasn't worked on this and isn't ready to discuss it.

AI12-0208-1/00 Predefined bignum support

Erhard asks if John's Bignum package even was submitted. Not yet (Jeff has a copy but isn't sure he has permission to distribute it). Ed notes that every Ada compiler has a similar package. Someone chimes in that “so does Codepeer”.

Steve discusses “Round” function, for dealing with capacity limits. The idea is that if the number doesn't fit into the capacity, one could use Round to give some value nearby. This is too weird; one would want the parameter to be something meaningful to the user (rather than capacity).

Bob doesn't think we need Bounded Rational at all. It seems unusual, as the usual desire is exact results. Ed concurs. We're not certain, but we should at least consider not having a Bounded Rational package, especially if Rounding gets too screwy.

Tucker thinks that users can write Round if they need it, using the Numerator and Denominator of the values.

AI12-0242-1/02 Reduction expressions

We are almost out of time. A quick look at this AI. Tucker suggests that we've at least agreed on an attribute syntax. Aggregate'Reduce. Randy notes that a bare aggregate is not currently allowed as an attribute prefix, we'll have to allow that explicitly (but since qualified names are currently allowed, it seems unlikely to be a major problem).

Jean-Pierre doesn't like having 'Reduce following the expression. He says that it can get lost after a large aggregate expression, and doesn't like the implication that the aggregate gets evaluated. Randy notes that it *might* be evaluated, as a reduction on an array object makes perfect sense (or even a function call as the prefix).

Jean-Pierre suggests switching the operands and having the attribute return a function. That seems weird, and one would not want to materialize such a function, either, so its not clear what's being gained.

We're out of time, so it is suggested that Jean-Pierre should write some sort of alternate proposal which we can consider by e-mail.

AI12-0243-1/01 Subtypes as primitive arguments

One approach would be to require static matching for the subtypes of controlling parameters. That would require extending static matching to subtypes of derived types.

Bob suggests that we should solve this problem through a class-wide type.

Steve thinks that we can make subtype matching work.

Steve will take the AI.

Keep alive: 5-1-5. Jeff voted against. Gary originally voted against, then changed his vote to abstain.

AI12-0244-1/01 Check name for Value attribute

No one volunteers to check the index for other errors. Bob opines that it is a waste of time. We decide to forget that and fix any problems as they are discovered (essentially, what we've been doing for the last 23 years).

“base type of the attribute” doesn't make sense. Change the wording to:

> ...base {sub}type of {the prefix of} the [attribute]{attribute_reference}.

Approve AI with changes: 11-0-0

AI12-0245-1/01 “with and use” clause

Jean-Pierre says that use clauses should be used in the inner-most scope where they are needed, so the number of them globally should be small. (This is discussed in the e-mail in the !appendix of the AI – Editor).

Randy notes that child packages work differently for **with** and **use**.

```
with A.B.C;   -- Means with A, A.B, A.B.C;
use A.B.C;    -- Means just this.
```

Tucker says this makes him uncomfortable.

We discuss whether there would be enough support to continue. Randy suggests putting a vote of support/opposition on the record. There isn't any support for that, either.

No Action: 9-0-2.

AI12-0246-1/01 Fixed first indices for variable-length arrays

Tucker says that he wanted this from day 1. The bounds are a tripping hazard for most uses.

Jean-Pierre says that it is a tripping hazard only if you don't know how to write in Ada. When writing in Ada, you have to be aware of bounds.

Steve thinks this would introduce a new tripping hazard. Because the bounds would change. Steve gives an example of renaming a slice and then indexing. Tucker thinks that this example is extremely unlikely.

Randy complains about the need to define “semi-constrained” subtypes to have this.

Tucker would prefer to have this part of the type; not a separate subtype. He thinks that avoids most of the oddities. Randy notes that wouldn't work for operations on existing string types, one can't change their declaration. [One wonders how applying a constraint to such a type would work – Editor.]

Tucker will take this.

Keep Alive: 8-0-3.

AI12-0247-1/01 Potentially blocking goes too far for Detect_Blocking

Randy explains the AI.

Approve AI: 11-0-0.

AI12-0248-1/01 Null array and empty container aggregates

Jean-Pierre suggests that (null record) exists and null arrays should work similarly if we're going to have a special mechanism for them.

Erhard notes that he has often used empty strings, but he does not recall to have ever needed an expression for an empty array. To add syntax for a feature hardly ever used seems very wrong.

Steve suggests an attribute T'Empty. That would provide the empty object for any composite type. Randy wonders why we want that, he thought the idea was to be able to write an aggregate. The containers all have Empty_xxx constants already.

Jean-Pierre suggests that the container case is very different. He would like them separated. Several people agree.

Raphaël suggests that whatever is chosen should be the same for every kind of type. The idea is to unify the model. Tucker notes that we want arrays and containers to work similarly. Several people agree with that.

Bob objects to the lower bound change for null arrays. Tucker argues that the current rule prevents null arrays of type indexed by modular types.

Straw poll: Should we fiddle with the rule for 'First for the null string? Leave alone: Bob, Ed, J-P; Definitely change: Tucker; Steve; Raphaël; Abstain: rest. There is not enough consensus to do this.

Tucker gets the AI.

Keep alive: 8-1-2. Bob is against. Tucker, Erhard, abstain.

AI12-0252-1/01 Duplicate interrupt handlers under Ravenscar

Attachment of a nested object over some other attachments feels like Dynamic Attachment to the group, so make this rule apply just to that restriction alone. Randy notes it's still weird if nested units have handlers (that still is rather dynamic); No_Dynamic_Attachment doesn't make a ton of sense without No_Local_Protected_Objects. (And even with it, it allows dynamic interrupt numbers, which is also weird.) Regardless, we'll just fix this issue.

> If [both] restriction No_Dynamic_Attachment [and restriction No_Local_Protected_Objects are]{is} in effect, then a check is made that the interrupt identified by an Attach_Handler aspect does not appear in any previously elaborated Attach_Handler aspect; Program_Error is raised if this check fails.

Approve AI with changes: 10-0-1

AI12-0253-1/01 Saturation arithmetic for scalar types

Randy argues that this should be a set of packages. With the numeric literal fix, a set of generic packages have almost the same capabilities as built-in math.

Do we need to standardize these packages? There are some fields where this sort of math is heavily used. It could open new uses for Ada.

We are going to focus on choice (3), the other choices are too heavyweight.

Ed will take the AI; Raphaël will help with this.

Keep alive: 11-0-0.

AI12-00254-1/01 Bounded_Indefinite_Holders

We briefly discuss the idea.

We look at the options. Tucker agrees that we want all of the containers. He notes that we'll need something like a Bounded_Indefinite_Container if we have users that need predictable finalization and predictable memory usage.

Raphaël thinks this is overkill. He notes that you can use Holders as elements as needed. Several people worry about combinatorial explosion of containers.

Jean-Pierre notes that “Bounded” is somewhat a misnomer. For most Bounded containers the bound is on the number of elements; for the Bounded_Indefinite_Container, the bound is on the size of an individual element.

We briefly discussion proposed aspect No_Controlled_Subcomponents. Several people question the need, suggesting that the global restriction should be good enough. [Editor's note: We don't have a specific global restriction, but someone could use No_Dependence (Ada.Finalization) for this purpose.] A global restriction prevents use of other Ada features like Storage_Pools, which might be too much restriction for an application. So No_Controlled_Subcomponents should be a separate AI.

Randy will take the AI and write just Bounded_Indefinite_Holders.

Keep alive: 8-0-3 (Gary, Bob, Ed abstained)

AI12-0255-1/01 Component iteration/reflection

Raphaël studied the problem; it wasn't practical to make the changes to support compile-time reflection.

Tucker notes that one can't write something like streaming yourself. Randy notes that you can write an interface that would let you look for a particular type of component, but nothing for components of unknown types.

No Action: 8-0-3 (Tucker, Brad, Bob abstained).