Minutes of Electronic ARG Meeting 58B

5 March 2018

Attendees: Raphaël Amiard (until 12:20 EST), Steve Baird, Randy Brukardt, Jeff Cousins, Gary Dismukes, Bob Duff, Brad Moore, Erhard Ploedereder, Jean-Pierre Rosen, Ed Schonberg, Tucker Taft, Tullio Vardanega.

Observers: Peter Chapin, Justin Squirek.

Meeting Summary

The meeting convened on Monday, 5 March 2018 at 10:35 hours EST and adjourned at 13:35 hours EST. The meeting was held using Zoom. The meeting covered just a portion of the oversized agenda, but did cover all of the regular AIs.

AI Summary

The following AIs were approved with editorial changes:

AI12-0254-1/02 Bounded_Indefinite_Holders (7-1-4)
AI12-0258-1/01 Containers and controlled element types (11-0-1)
AI12-0259-1/01 Lower bound of strings returned from Ada.Command_Line (12-0-0)
AI12-0260-1/03 Functions Is_Basic and To_Basic in Wide_Characters.Handling (11-0-0)
AI12-0261-1/01 Conflict in “private with” rules (7-0-5)

The following AIs were approved for intent, but they require a rewrite:

AI12-0075-1/07 Static expression functions (12-0-0)
AI12-0079-1/06 Global-in and global-out annotations (8-1-3)
AI12-0119-1/07 Parallel operations (10-0-1)

Detailed Minutes

Previous Meeting Minutes

No one has any changes to the minutes. Approve minutes: 12-0-0.

Date and Venue of the Next Meeting

The next meeting is proposed for Monday, April 2th, at 11:00 EDT. It is noted that is a holiday in some countries (Easter is the previous day). We check other dates, but they work worse for various people. We end up returning to the original date.

ACATS Tests

Jean-Pierre created a Doodle poll to sign up volunteers for the ACATS tests. See it at https://doodle.com/poll/f862izpz9nyp2y22. At this time, we'll just look for volunteers; people that have a lot of homework probably shouldn't volunteer now.

For now, just volunteer on the Doodle poll, we'll revisit in April to see if enough people are volunteering.

Prioritization of Amendment AIs

Randy has made a color coded map of all of the Amendment AIs proposed for Ada 2020. Find it at http://www.ada-auth.org/ai-files/grab_bag/2020-Amendments.html.

We have to submit a scope (which usually is a list of AIs that we intend to include) to WG 9 for their June meeting. So it will be critical to cut this list to something manageable.

The map shows the relationship of the many AIs to the WG 9 instructions (and also links to each AI in case you've forgotten what it is about). In general, the AIs that directly answer the WG 9 instructions should have the highest priority (with a few exceptions, like [pun alert] the exception contracts waiting for the SPARK take on that).

We need to determine what priority to give to the rest of the AIs. Randy proposes using a Letter Ballot for that purpose, so that everyone gets an equal say – as opposed to just the management and a few other voices. He suggests that each ARG member ranks their top-ten AIs of the unfinished ones.

We are supposed to have a mostly finished draft to circulate next March -- only about 12.5 months from today.

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
- AI12-0210-1 (get guidance from the SPARK group, then update the AI)
- AI12-0243-1

Randy Brukardt:

- AI12-0017-1 (assist Florian Schanda)
- AI12-0112-1
- Check all 10646 references and update any needed to 10646:2017 (see discussion of AI12-0260-1; assigned AI12-0263-1 after meeting)

Editorial changes only:

- AI12-0254-1
- AI12-0258-1
- AI12-0259-1
- AI12-0260-1
- AI12-0261-1

Alan Burns:

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

Brad Moore:

- AI12-0119-1
- 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:

- AI12-0075-1 (polish wording)
- 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

AI12-0075-1/07 Static expression functions

Move “only” in the Legality Rule:

Aspect Static shall [only] be specified to have the value True {only} if the associated expression function:

The bullets don't quite fit with the lead-in sentence. It seems unusual in any case. Randy notes that he rewrote this provisionally, noting that it “needed wordsmithing”, but that was the end of the conversation.

Tucker volunteers to rewrite this paragraph.

The term should be “type-invariant enforcing”. Gary approves, so this **must** be necessary (he's always removing hyphens). Everyone laughs.

Approve intent: 12-0-0.

AI12-0079-1/06 Global-in/global-out annotations

Tucker tries to explain Reachable. He goes on to mention other changes, including the definition of freezing for library units.

Steve notes that SPARK has a similar issue with the Initial_Condition aspect on a library unit.

Tucker wonders about configuration pragmas that follow a library unit. For instance, a pragma Inline or Convention; these ought to work on a subprogram unit, but they can't if the unit is frozen before they're applied. Randy jokingly says that they should have used an aspect.

Erhard wonders if Reachability can be determined statically. He's afraid that it would require some sort of dynamic closure. Tucker explains that Reachable works on types, not dynamic pointers.

Class-wide operations seem to cause problems, as one doesn't know the components (and thus the types) involved with any extensions. (There might also be an issue with statically tagged items when the actual tag is some other type, although that seems less likely.)

Approve intent of AI: 8-1-3

Erhard votes against as he thinks that it is a quagmire to try to get static checks on heap access. He thinks that there is value to statically global variables. Ed notes that the existence of Global as a contract has the useful effect of minimizing the use of such global variables.

AI12-0119-1/07 Parallel operations

Ed: “Add after 5.5(9/4):” end with “mentions the loop parameter sequential.” What does this mean? Brad says he's not sure, he'll have to fix it.

Ed notes that some algorithms are safe to parallelize even though they look like that have dependences, so we need to be able to turn off the checks. He gives an example of Lagrangian Relaxation, where the values of elements depend on “nearby” elements. [And I thought “Lagrangian Relaxation” was LaGrange's goal when he went on vacation. ☺ - Editor.]

Tucker would like to split the data race checks into a separate AI. That would address the possibility of turning them off (or on), so that users could simply assert that their code is OK.

Erhard makes the point that the static check for Nonblocking is a similar problem. Randy notes that that could be a performance problem – specifically, if blocking is to be supported, then a supposedly lightweight tasklet would need the whole mechanism for blocking (queues, delays, etc.).

There is some discussion of whether blocking blocks the entire task (and possibly all tasklets) or just one. Tucker notes that it is important to know what happens when a tasklet blocks. Having it totally unspecified gives users and implementers no guidance as to what they can expect to work. Steve starts describing a loop that has one iteration blocking and waiting for another iteration. Gack!

Erhard says that blocking should just be erroneous. That seems too unspecified. Tucker suggests to treat this like being in a protected operation, with potentially blocking operations being a bounded error. One could even use the same runtime mechanism (including Detect_Blocking). Static checks can then be supported using Nonblocking (optionally).

Steve wonders why the wording treats exceptions differently than other transfers of control. (The wording specifically calls out exceptions.) Brad thinks it might be an artifact of older rules where exceptions were handled with a priority over other exits. Steve worries about abort being a transfer of control; he worries that it could cause the rules to be recursive. (This idea makes your editor's head hurt.)

Gary notes that it would help to mention exceptions somehow (as redundant, expository text), lest readers be left wondering how exceptions are handled.

Jean-Pierre notes that the Legality Rules under parallel blocks says “to the loop”. It ought to be talking about a block.

Tucker would like to lock down the syntax. Do we still like the syntax as proposed?

Is **parallel** reserved? Yes, that's in the AI. Someone brings up non-reserved keywords again; Tucker calls it the third rail of ARG work (a pretty accurate description). We're not going there.

Jeff: 5.5(6/5) has “reserve{d}” (missing a 'd') in two places.

Tucker argues that **end do** is unusual. Other uses of **do** end with some other word. Someone suggests **end parallel**. **Parallel** isn't a kind of statement, though, its more of a modifier. There doesn't seem to be much interest in making a change here.

Brad noted that the chunking code (D.16) should probably be related to AI12-0251-1; Tucker suggests creating that as an alternative to AI12-0251-1. (It's not completely alternative.)

Steve notes that doing a **goto** to the end is different than falling through. Go to means “goto now”, not when everything is done. Several people are confused about what he is talking about, it is an example like:

```
parallel do
    A := Fun(1);
and
   B := Fun(2);
and
   goto After;
end do;
<<After>> null;
```

Randy suggests splitting the parallel iterator stuff into a separate AI. We want to make this AI as small as possible. We want parallel iterators, but they're not fundamental to the parallel loop/block features. The group agrees.

So split into four separate AIs (we already split this twice before):

- Manual chunking stuff (alternative to AI12-0251-1);
- Parallel container iterators (new AI);
- Data race/nonblocking checks (new AI);
- And the remaining basic definition of parallel blocks/loops (in AI12-0119-1).

Brad will split these, and then next time we'll look at re-assigning some of them to others.

Steve wonders if “request” is used this way in the Standard [I didn't hear or record where he was reading]. If not, the wording should be changed to be more typical.

Approve intent: 10-0-1.

AI12-0254-1/02 Bounded_Indefinite_Holders

Steve: The precondition is backwards, the relational operation should be “>” instead of “<=”.

Tucker: In the Bounded Errors portion: “It is a bounded error {to} call...”

In the Implementation Requirements: there is a double underscore in Indefinite__Holders.

The Bounded Error is weird, since it is not a user mistake. This should be written as an Implementation Permission to raise Program_Error if this not supported.

Even as an implementation permission, this is still a portability problem as users can't know if their types are discontiguous. It's possible to implement discontiguous types (at an additional cost) – that doesn't work for other kinds of Bounded_Indefinite_Containers, but we aren't defining those now.

So remove the first paragraph of the bounded error (and the two notes).

Jeff: Typo in [1] in the !discussion: “decallocation”.

Gary: There is a typo in the !summary “[are]{is} not allowed”.

Tucker: There is a spelling error in the penultimate paragraph in !discussion: should be “targeting”, not “targetting”.

Gary wants to remove hyphens from “over[-]estimation” and “run[-]time” in the !problem. The first paragraph should “static[-]{ }analysis”.

Jeff: 3rd para of !problem “array” in singular and “records” in plural, should be “arrays”.

Approve AI with changes: 7-1-4. Ed voted against as he thinks this is too low priority to put into the language. Tucker disagrees, he often wants to have static max size for class-wide types. There is more discussion, with a couple of people offering to change their votes to ensure passage (the above vote is passing according to our rules, so that wasn't necessary).

AI12-0258-1/01 Containers and controlled element types

Jeff: Last line for wording, “convinient” is spelled wrong, should be “convenient”.

Tucker suggests: “...is not {an} indefinite {container}.”

Bob notes that C++ does this better because it has “placement new”. But we don't have that. Bob thought about proposing it, and then thought better of it thinking about how much work it would be. We agree that adding something like that would be a huge issue.

... when it is deleted {from the container} (or when the container object is finalized if the element {has not been}[is not] deleted) ...

Gary: !discussion, last paragraph. “..parts [to]{in} their element types...” “...concerns can use[s] a Bounded_Indefinite_Holder ...” “These suggest that {users}[a user] who have...”

Steve wonders about the sloppiness of the name; we're talking about the name of the generic unit, not of the instance.

“...based on a generic whose name...” (two places). Or maybe “...an instance of a generic whose name...” Leave this to the editor.

Approve AI with changes: 11-0-1.

AI12-0259-1/01 Lower bound of strings returned from Ada.Command_Line

In discussion: “langauge”.

Steve asks about the bounds of the null string. The wording seems to allow the bounds to be 42 .. 12. This seems irrelevant, any attempt to index the null string will raise Constraint_Error regardless of the bounds. We don't care WHY the bounds fail.

Approve AI with changes: 12-0-0.

AI12-0260-1/03 Functions Is_Basic and To_Basic in Wide_Characters.Handling

Would anyone use this? It seems likely that it would be used. Jean-Pierre notes that French sorting ignores accents. If one is using Unicode and wants to implement such a sort, these functions are needed. (This is not purely academic, he raised the question after someone asked on fr.comp.lang.ada how to do this.)

Should we update to 10646:2017? Tucker thinks we should to stay up to date. Randy notes that there other references in the Standard that would need to be updated.

Randy will make a separate AI if there are any other normative references that need to be changed.

Gary: “purpose[d]ly”.

Approve AI with change: 11-0-0.

AI12-0261-1/01 Conflict in “private with” rules

Randy notes that he and Tucker worked out better wording on Friday evening; he proposes that we use that wording.

Gary: !summary, should say that “Only one of the bullets ... need{s} to be True.”

Approve AI with changes: 7-0-5.

Several people note that they abstained because they are not certain of how the rewording fits together.