Minutes of Electronic ARG Meeting 62C

29 April 2020

Attendees: Raphael Amiard, Steve Baird, Randy Brukardt, Jeff Cousins, Gary Dismukes, Claire Dross, Bob Duff, Brad Moore, Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

Observers: Arnaud Charlet, Erhard Ploedereder (joined 11:50 AM EDT).

Meeting Summary

The meeting convened on Wednesday, 29 April 2020 at 10:34 hours EDT and adjourned at 13:30 hours EDT. The meeting was held using Google Meet. The meeting covered many of the AIs on the agenda.

AI Summary

The following AIs were approved with editorial changes:

AI12-0205-1/04 Defaults for generic formal types (10-1-3)
AI12-0282-1/05 Atomic, Volatile, and Independent generic formal types (11-0-3)
AI12-0366-1/02 Changes to Big_Integer and Big_Real (13-1-0)
AI12-0373-1/05 Bunch ‘o fixes (14-0-0)
AI12-0375-1/01 Meaning of Global when there is no mode (13-0-1)
AI12-0376-1/01 Representation changes finally allowed for untagged derived types (14-0-0)

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

AI12-0362-1/01 Floor and other rounding attributes for fixed point types (13-0-1)
AI12-0363-1/03 Fixes for Atomic (12-0-2)
AI12-0374-1/01 Fixes for Nonblocking (14-0-0)

The following AIs were discussed and assigned to an editor:

AI12-0346-1/02 Ada and OpenMP
AI12-0377-1/01 View conversions and out parameters passed by copy revisited

The following AIs were discussed and voted No Action:

AI12-0079-1/13 Global-in and global-out annotations (14-0-0)
AI12-0079-2/01 Global-in and global-out annotations (14-0-0)
AI12-0240-6/05 Global aspect and access types used to implement Abstract Data Types (14-0-0)
AI12-0303-1/03 Some constants must be covered by Global aspects; extensibility (14-0-0)
AI12-0310-1/03 Specifying private parts of packages in aspect Global (14-0-0)
AI12-0334-2/04 Predicates and Global/Nonblocking (14-0-0)
AI12-0353-1/01 Global and Nonblocking aspects and dispatching calls (14-0-0)

The following AIs were discussed and placed on hold:

AI12-0243-1/03 Subtypes as primitive arguments (14-0-0)
AI12-0243-2/01 Subtypes as primitive arguments and class-wide predicates (14-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

None.

Previous Meeting Minutes

No one has any changes to the minutes (all previously sent suggestions have already been applied). Approve minutes: 14-0-0.

Date and Venue of the Next Meeting

Our next electronic meeting is scheduled for June 13, 10:30-1:30 EDT; this is the time slot of the second day of the canceled in-person meeting. (WG 9 has the first day). This is a Saturday. No one voices any objections.

Ada 202x Scope

Towards the end of the meeting, we have a brief discussion on the scope of the language.

Arnaud is concerned about "squeezing too much" into Ada at the last minute. It is noted that the AI that prompted this discussion has a small number, indicating that it was part of the original work scope of Ada 202x.

We're not accepting new submissions from anywhere other than ARG members and AdaCore, and even that stops as of June 1.

We look at the remaining AIs on the agenda after completing discussions for the day, and note that most of them relate to fixes or improvements to features already proposed for Ada 202x. The only one that isn't related to an existing Ada 202x feature was an old proposal that was reactivated by a request from some AdaCore employees. Ergo, the scope of Ada 202x is essentially what it is today, there's very little new that remains to be discussed.

Unfinished Action Items

There are three unfinished action items (Steve Baird, AI12-0016-1; Ed Fish: AI12-0215-1; Tucker Taft: Help editor with contracts). We did not spend any time talking about these.

Current Action Items

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

Steve Baird:

- AI12-0016-1
- AI12-0362-1
- Talk to the WG 9 chair about the possibility of a Technical Report on usage of Open MP with Ada (see discussion of AI12-0346-1).

Randy Brukardt:

- AI12-0378-1 (split from AI12-0377-1 at this meeting)

Editorial changes only:

- AI12-0205-1
- AI12-0282-1
- AI12-0366-1
- AI12-0373-1
- AI12-0375-1
- AI12-0376-1

Edward Fish:

- AI12-0215-1 (with help from Raphael Amiard)

Tucker Taft:

- Review AI12-0302-1 now that AI12-0079-3 is approved and make any needed changes.

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.
- Review the Nonblocking model, including the fixes of AI12-0374-1.
- AI12-0377-1
- Help the Editor convert language-defined preconditions to the form recommended by AI12-0370-1.
- Help the Editor update the containers Global aspects (see AI12-0079-3).

Richard Wai:

- Review the Nonblocking model, including the fixes of AI12-0374-1.

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 24 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

AI12-0079-1/13 Global-in and global-out annotations
AI12-0079-2/01 Global-in and global-out annotations
AI12-0240-6/05 Global aspect and access types used to implement Abstract Data Types
AI12-0303-1/03 Some constants must be covered by Global aspects; extensibility
AI12-0310-1/03 Specifying private parts of packages in aspect Global
AI12-0334-2/04 Predicates and Global/Nonblocking
AI12-0353-1/01 Global and Nonblocking aspects and dispatching calls

Last time, we approved AI12-0079-3 to replace the existing global stuff. What we didn't do is eliminate the four approved AIs that made up the previous global proposal (AI12-0079-1, AI12-0240-6, AI12-0303-1, AI12-0310-1), the two pending fix AIs for that proposal (AI12-0334-2, AI12-0353-1), and an unused alternative to that proposal (AI12-0079-2). There are some parts of those AIs that are still needed; all of those have been moved to other AIs on today's agenda.

We decide to vote on these as a group.

No Action: 14-0-0.

AI12-0205-1/04 Defaults for generic formal types

Randy reverted to a simple rule, as it matches what is done for generic formal subprograms, it is useful for the known use cases, and we can do more in the future if possible. It seems messy to do more (see the discussion), and it seems that doing more should be done consistently (for all defaults, both new and existing).

Jean-Pierre has a typo: "a[n] generic actual".

Jeff: Discussion, 9th paragraph, "Any instance will [be] detect the error."

"A[n] test".

Gary: Last paragraph of discussion "to use a [a] more ..."

Tucker: in Example: "inadvertantly" → "inadvertently"

Raphael is concerned about the syntax of this change. There is concern that the syntax of generic formals is hard to remember. (This came from AdaCore users.) Most Ada syntax is regular enough to be easily remembered. Richard notes that there is a lot of things that you can't remember in Ada, because it is a large language. It's easy enough to look these things up. (Or ask your IDE for help.)

Approve AI with changes: 10-1-3. Opposed: Raphael (for reasons noted above); Abstain: Claire, Steve, Justin.

AI12-0243-1/03 Subtypes as primitive arguments
AI12-0243-2/01 Subtypes as primitive arguments and class-wide predicates

We did not find a good solution for these. The simple solution has problems with dispatching routines that are statically bound (need a mechanism like Pre'Class), and the more complex solutions are, well, complex. Tucker wonders if we can live without this at all, since the functionality can be provided less conveniently with Pre'Class.

Hold: 14-0-0.

AI12-0282-1/05 Atomic, Volatile, and Independent generic formal types

This fixes a serious compatibility problem. **limited private** should match as much as possible, including volatile and atomic objects. Otherwise, it becomes hard to create a "universal" generic unit.

Tucker separated the rules into rules about objects and rules about types. Thus, he rewrote and reorganized C.6(12) and C.6(12.1).

We look at the rule for volatile arrays. We decide to leave it alone, since it is existing and dates back quite a few years.

Tucker explains the Implementation Permission. Randy asks if we need a corresponding rule for volatile objects. The atomic rule also applies to volatile. So add "A corresponding permission applies to volatile parameter passing."

Steve wonders about elementary objects associated with the permission. This doesn't change anything for that. We decide to add "subprogram with a parameter of type T {that is normally passed by reference}."

Gary: The first paragraph of discussion, second instance of Atomic_Components is misspelled.

Approve AI with changes: 11-0-3. Jean-Pierre, Jeff, Raphael abstain.

AI12-0346-1/02 Ada and OpenMP

Tucker is proposing writing a Technical Report to describe such a mapping. There would be no normative words at this point.

Tucker is volunteering to write this technical report.

We probably should keep the AI open in case some Standard wording adjustments are needed.

Keep alive: 14-0-0.

Should we bring this up to WG 9 as a potential work product? It's probably too soon to make an official work item, but WG 9 should be advised of the plan for this Technical Report.

AI12-0362-1/01 Floor and other rounding attributes for fixed point types

Do we want to do this now in Ada 202x? Jean-Pierre raised this. He is asked to elaborate. He says that other people asked for this on comp.lang.ada. It is noted that fixed point is underused, and improving the capabilities should help.

It is suggested that implementations can always provide additional attributes. Randy notes that they can't use these names, since they're already language defined. Randy reads 4.1.4(12/1): "An implementation may provide implementation-defined attributes; the identifier for an implementation-defined attribute shall differ from those of the language-defined attributes unless supplied for compatibility with a previous edition of this International Standard."

We could add a permission to allow these to be defined. That seems to be enough for Ada 202x, and we can revisit these in the future. Steve will take the AI.

Approve intent: 13-0-1. Bob abstains.

AI12-0363-1/03 Fixes for Atomic

Randy notes that there is new discussion on Ada-Comment, so it is premature to approve this one. Tucker says that we still should talk about this.

Randy notes that hardware that malfunctions when accessed with the wrong size is fairly common, at least according to the responses on Ada-Comment. We need to be responsive to the original request in AI12-0128-1. He notes the original request was for a dedicated aspect, and while that led to suggesting adding additional requirements to Atomic, that requires the programmers to take extreme care. The Ada way is more to tell the compiler what is required.

Richard notes that writing machine register access in Ada code using components that are mapped to particular bits results in much more readable code than the traditional bit masking code.

We think that we need a separate AI to add an aspect to require exact size access.

And then this AI will allow subcomponents of atomic types and objects to be atomic.

Randy notes that he doesn't like Tucker's proposal to change 'Access for this unusual case. It adds complexity to the core for an obscure corner case of an Annex feature. We should keep annex features out of the core as much as possible.

Approve intent: 12-0-2. Bob, Gary abstain.

AI12-0366-1/02 Changes to Big_Integer and Big_Real

There isn't a way to explicitly create an invalid object, but an uninitialized object still is invalid.

Tucker uses Valid_Big_Integer in order to avoid using preconditions everywhere to check validity.

Tucker also made several wording improvements.

In A.5.7, `use all type Big_Integers;` should be `use all type Big_Integers.Big_Integer;`

Tucker didn't change the postcondition of Denominator. Bob would like a complete postcondition.

```
(if Arg = 0.0 then Denominator'Result = 1 else
  Greatest_Common_Divisor (Numerator (Arg), Denominator'Result) = 1);
```

Also, `Post => (if Arg = 0.0 then Numerator'Result = 0)` on Numerator. We can't mention Denominator here, as that would be recursive (since the Post of Denominator calls Numerator).

Someone suggests that the Post of Denominator contain the complete Post for Numerator as well, since it already has most of it. But that seem too weird. Perhaps an AARM note would clarify.

Arnaud notes that Big_Positive and Big_Natural need to allow invalid values:

```
Dynamic_Predicate => (if Is_Valid (Big_Positive) then Big_Positive > 0),
```

A similar Dynamic_Predicate is needed for Big_Natural.

Bob notes that Put_Image doesn't match what GNAT implemented. That's a separate AI yet to be discussed or even agreed on.

Gary: Typo, last paragraph of !discussion: "demonator". Randy notes that this whole statement is now false, so it needs rewriting or removal.

The function result should be Valid_Big_Integer for the two definitions of To_Big_Integer.

Approve AI with changes: 13-1-0. Bob is opposed (he does not like the names, he specifically does not like saying valid all over the place; he also notes that he thinks that he lost on the names a while ago).

Later, Arnaud says that he has two other problems with this AI. He notes the definition of Valid_Big_Integer is wrong:

```
subtype Valid_Big_Integer is {Big_}Integer ...
```

That seems pretty important.

His other problem is that the definition of From_String doesn't statically match for the user-defined literal definition. But that is not required by the rules of 4.2.1. And that was intentional, we want to be able to allow functions returning any subtype of the specified type. There's no reason to assume that a literal can represent every possible value of a type. So there is no problem here with that.

We agree to include the change into the AI without a new vote.

AI12-0373-1/05 Bunch 'o fixes

(7) Tucker suggests that the list of representation aspects in the AARM be updated. Randy wonders if we should keep maintaining this list. Arnaud says that that list is handy, so we should spend the time to update it rather than getting rid of it.

The editor will take care of that.

Gary: Typo in Discussion of (7) "the usage of Bit_Order is clear{ly} that of...".

Jeff: Typo in Discussion of (3): "rename {an} array subcomponent...".

Approve AI with changes: 14-0-0.

AI12-0374-1/01 Fixes for Nonblocking

There are some extra blank lines in some places. Jean-Pierre says that some of the line feeds are Unix and some are Windows.

Erhard questions the summary item:

(1) Generic units that are declared Nonblocking use the "and" of all of the formal parameters as their nonblocking expression.

We discuss this for a while and eventually decide that it is correct, since the rules modified are actually on the generic unit. Randy says he did that in order to avoid having to revisit all of the Legality Rules for use of calls inside of a generic unit – these are rather difficult to get right and making them even more complicated didn't seem to help anything.

Jeff notes that 9.5(49/5) is duplicated under both (2) and (3). The copy under (3) should be removed. [Editor's note: It is subtly different than the copy under (2), because it needed to be changed for (3). I added a note to that effect so that the extra changes are noted in case some part of this AI is dropped.]

Richard says that he would like to review Nonblocking, and Tucker also will do so.

Approve intent: 14-0-0.

AI12-0375-1/01 Meaning of Global when there is no mode

We couldn't agree on what the default should be, so we eliminated the default.

Brad asks if global_group_designator is used anywhere. After some work, we decide it is not, so it should be removed completely.

Approve AI with changes: 13-0-1 (Bob abstains).

AI12-0376-1/01 Representation changes finally allowed for untagged derived types

Randy notes that there is no problem with other rules, with the possible exception of the out parameter rules (which we are revisiting anyway).

Erhard is confused by the wording. We agree to clarify "it" by introducing a name:

> It is illegal to specify a nonconfirming type-related representation aspect for an untagged by-reference type T if it is derived from a by-reference type, or if one or more types have been derived from T prior to the specification of the aspect.

Brad has a typo: discussion in 2nd paragraph: "… part about primitive subprogram{s} ..."

Approve AI with changes: 14-0-0.

AI12-0377-1/01 View conversions and out parameters passed by copy revisited

Claire worries that this rule does not fix the entire problem. There is a discussion of the dynamic check. That only occurs in generic bodies and it is the usual rule to avoid contract model problems. In SPARK it is a compile-time error, and in GNAT there is always a warning on generic expansion.

Split the AI into scalar and access parts. The access case is about erroneous execution.

Randy will take the access part [assigned AI12-0378-1 after the meeting]. Tucker will take the scalar part (which will remain AI12-0377-1).

Keep alive: 14-0-0.