Minutes of Electronic ARG Meeting 60F

9 April 2019

Attendees: Steve Baird, Randy Brukardt, Jeff Cousins (joined 11:11 EDT), Gary Dismukes, Bob Duff, Brad Moore , Erhard Ploedereder (joined 11:40 EDT), Jean-Pierre Rosen, Ed Schonberg, Justin Squirek (left 13:50 EDT), Tucker Taft.

Observers: Richard Wai.

Meeting Summary

The meeting convened on Tuesday, 9 April 2019 at 11:04 hours EDT and adjourned at 14:01 hours EDT. The meeting was held using Zoom. The meeting covered some of the AIs on the agenda.

AI Summary

The following AI was approved:

AI12-0298-1/05 Revise the conflict check policies to ensure compatibility (7-0-4)

The following AIs were approved with editorial changes:

AI12-0304-1/01 Image attributes for language-defined types (10-0-0)
AI12-0313-1/02 Introduction for Ada 2020 (11-0-0)
AI12-0324-1/02 More changes from the RM review (10-0-0)
AI12-0325-1/02 Various issues with user-defined literals (10-0-0)
AI12-0327-1/02 Clarify iterator filter wording for aggregates (11-0-0)
AI12-0329-1/01 Naming of FIFO_Streams packages (9-0-2)

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

AI12-0326-1/01 Consequence for incorrect Allows_Exit (7-0-4)
AI12-0328-1/01 Meaning of limited type and record type in 4.5.2(28.1/4) (7-0-3)

The following AIs were discussed and assigned to an editor:

AI12-0240-6/01 Local access types for Abstract Data Types
AI12-0330-1/00 Add items to the Glossary

Detailed Minutes

Welcome

We welcomed each other as we joined the meeting.

Apologies

Tullio Vardanega said that he had a conflict and might not be able to attend (he didn't). Jeff Cousins reported that he might be late because road construction could delay his arriving home (it did).

Previous Meeting Minutes

No one has any changes to the minutes (all previously sent suggestions have already been applied). Approve minutes: 9-0-0.

Date and Venue of the Next Meeting

Randy had suggested April 30th. We tried many dates from then through mid-May. We finally settled on May 9th, at the usual time of 11:00 EDT.

Our next face-to-face meeting is scheduled for June 14-16 in Warsaw Poland, immediately after the Ada-Europe conference.

Unfinished Action Items

Steve Baird has the only unfinished action item, the "dynamic accessibility is a pain to implement" AI (AI12-0016-1). We did not spend any time talking about it this time.

Current Action Items

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

Steve Baird:

- AI12-0016-1

Randy Brukardt:

Editorial changes only:

- AI12-0304-1
- AI12-0313-1
- AI12-0324-1
- AI12-0325-1
- AI12-0327-1
- AI12-0329-1

Tucker Taft:

- AI12-0240-6
- AI12-0313-1 (one new bullet, see discussion)
- AI12-0326-1
- AI12-0328-1
- AI12-0330-1

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 19 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

AI12-0240-6/01 Local access types for Abstract Data Types

The idea is to recognize that there exist logical objects ("Compound objects") with a complex representation composed of multiple physical objects. A compound object can be treated (by clients) as a single entity, regardless of the parts of which it is made.

This is handled by marking access types as “local” access types. Each (allocated) object of a local access type belongs to exactly one compound object; it can be used in other ways as needed.

This is a minimal proposal intended to form a building block for future stronger schemes. It is intended to minimize assumptions.

Steve (playing Devil's advocate) asks why we need to do something now. Why not just do nothing? Tucker says that any ADT of any complexity is going to be implemented in part with pointers, and having no way to use those in parallel constructs is terrible.

Richard worries that this is too permissive. He would like to be able to force in the future a more restrictive model. Tucker hopes that future aspects would automatically set these aspects, essentially layering a stronger model on top of this one.

Erhard is worried that Compound would have to be specified on all types. No, just on the ones that are visible through the interface. The concept is recursive and reaches through as many objects as needed. The entire idea is to hide the implementation complexity behind single objects. Randy notes that Vectors would probably be the only compound type in the vector container. None of the other declared types would be compound.

Jean-Pierre notes that the ASIS section can be “no changes needed” so long as it is just a pair of aspects.

Randy asks about the term “local”. He notes that it adds confusion: “local access object” could be of a local access type, or a local object of some access type. Richard had suggested “subordinate” in e-mail. Ed suggests “secondary”, “restricted”. Brad says “managed”. Richard now suggests “Internal”. No conclusion is reached.

Tucker will take this back, with help from others as necessary.

Another meta discussion about whether there are any alternatives. Erhard suggests a purely implementation-defined mechanism. In that case, no one can write portable parallel code. Randy suggests that he would rather wait than put out a Standard without a portable way to write ADTs. The value of the parallel mechanism is the checking, and if we lose that, Randy starts to agree with Justin that the whole mechanism is insufficiently different than what we already have.

Approve intent: 5-2-4. Bob, Justin opposed; Gary, J-P, Jeff, Erhard abstains.

This vote fails. We try a “Keep Alive” vote instead.

Keep alive: 7-2-2. Bob, Justin opposed; Gary, Jeff abstain.

AI12-0298-1/05 Revise the conflict check policies to ensure compatibility

Tucker describes the changes. Parallel and Task constructs are separated, to avoid all compatibility issues. And the “known” checks are better defined, so they don't depend on dynamic rules.

There is some discomfort with the basic idea, but Tucker points out that we've already approved the idea, this is just a fix AI to correct unintentional incompatibilities. The feature isn't going away regardless of the vote on this AI.

Approve AI: 7-0-4. Bob, Erhard, Gary, Justin abstain.

AI12-0304-1/01 Image attributes for language-defined types

Bob wonders if there is a need to specify that Put_Image is not specified for a scalar type.

Tucker says that it isn't necessary, we don't allow arbitrary changes to language-defined packages.

Steve suggests that an AARM note is enough. Everyone agrees with that.

Gary notes that “non[-]scalar” shouldn't have the hyphen.

Approve AI with changes: 10-0-0.

AI12-0313-1/02 Introduction for Ada 2020

Randy wonders if the summary items are in any particular order. Some of the more important things are last.

Erhard complains that "arbitrary-precision integer" is nonsense. The precision of every integer is 1. Tucker starts reading from Wikipedia to show that this is a well-known term for Bignums.

Gary has some hyphen-changes. Third bullet under "Significant Changes": access{-}[ ]to{-}[ ]subprogram. Image attribute: "non[-]scalar". '@' "short[-]hand".

The Summary of Changes should say something about iterator filters.

Tucker will write a bullet for iterator filters.

"this 2020 edition" should be "this fourth edition"; we can't know what year this will be Standardized ahead of the fact.

Approve AI with changes: 11-0-0.

AI12-0324-1/02 More changes from the RM review

Gary notes that question (9) has "the the" at beginning of second sentence.

Gary is confused by (10) in the !wording. The replacement text isn't in the standard form; it should say "Replace <blah> with:".

Brad notes that the discussion has an extra word "...new sentence [with] uses...".

(8) wording: "{Evaluating} a delta_aggregate ..."

Approve AI with changes: 10-0-0.

AI12-0325-1/02 Various issues with user-defined literals

Jean-Pierre notes the item (2) in the !problem has a sentence that contains "one of the one of the".

Several people note that "convoluted" is spelled wrong in (4) in the problem: "rather convuluted descriptions".

Gary woud like one less comma in the wording for 4.2(10).

> The evaluation of a string_literal that is a primary {and has an expected type that is a string type[,]}

He also asked for a extra "that", Tucker objects as there are then too many "that"s in this text. Gary withdraws that suggestion.

There is an Author's Note with "echos" which should be "echoes".

Approve AI with changes: 10-0-0.

AI12-0326-1/01 Consequence for incorrect Allows_Exit

Tucker would prefer that is a Bounded Error. Bob wonders what are the bounds? We require a list of possible effects for a Bounded Error, unlike the case of Erroneous Execution.

Tucker thinks this is just Ada code. So if the exception is handled somewhere, the iteration might continue, or it might complete normally (rather than doing the return or goto expected).

The first wording sentence is messy.

Tucker will take a shot at this.

Approve intent: 7-0-4 Ed, Justin, J-P, Erhard abstain.

AI12-0327-1/02 Clarify iterator filter wording for aggregates

Jean-Pierre: "When we say" is not RM-speak. Tucker suggests dropping "when we say that"...

> [When we say that the] {A} sequence_of_statements of a loop_statement is {considered to be} *conditionally executed*, the statements are executed only when the filter of the iterator construct is satisfied.

"considered to be" is hardly any better. Try again (without the change markers):

> A sequence_of_statements of a loop_statement with an iterator construct is *conditionally executed*; that is, the statements are executed only when the filter of the iterator construct is satisfied.

This reads like is is defining semantics rather than just a term.

> A sequence_of_statements of a loop_statement with an iterator construct is said to be *conditionally executed* if the statements are executed only when the filter of the iterator construct is satisfied.

This still needs work, give to the editor. [Editor's note: Thanks a lot. I already spent 20 minutes trying to come up with sensible wording for this paragraph, and that was rejected right away.]

Gary: last paragraph of !problem: ... what [it]{is} actually...

Approve AI with changes: 11-0-0.

AI12-0328-1/01 Meaning of limited type and record type in 4.5.2(28.1/4)

Randy explains that Dynamic Semantics rules do not depend on views. So in the example given here, predefined equality is used (everywhere). Having the choice be view-specific leads to various anomalies, as the result of a membership could vary depending on whether it has visibility on the full type. (The full type in this example being elementary and non-limited, the predefined equality has to be used.)

Tucker worries that this is counter-intuitive. Several people note that it's also weird to get different results when the full view is visible – we can't win here, all choices are bad. Tucker suggests making those cases illegal. There is some concern about that, but Tucker notes that this configuration is rare – a limited private type with a defined equality, and that type is completed with something other than a record type or a immutably limited type.

We discuss why we don't use the primitive equality always. Tucker notes that there was a desire for X in 1 | 2 and X in 1 .. 2 to mean the same thing. If we used primitive equality for scalar types, they could mean something different.

Tucker will take the AI to come up with wording to make it illegal to use a membership when there is a limited partial view with equality. [Hopefully, one that depends on the completion as well – since there's no issue if the full type is limited type or a record type – Editor.] Randy notes that the Dynamic Semantics needs to be changed to talk about a "limited view", since Dynamic Semantics does not depend on views typically. Tucker is not initially convinced (noting that problem cases are illegal), but Randy notes that without doing that, the type wouldn't be considered limited anywhere, which would defeat the purpose of having the Legality Rule in the first place.

Randy notes that if we add normative wording, the AI will need to be rewritten into the form of a Binding Interpretation.

Approve intent: 7-0-3. Gary, Bob, Justin abstain.

AI12-0329-1/01 Naming of FIFO_Streams packages

We discuss the solution Randy proposed in this write-up. Randy notes that there was two options that had risen to the top in the e-mail discussion, and he prefers this one because it doesn't redundantly repeat “Streams” all over the place.

Tucker wonders why we have an abstract interface. Randy thinks that it was because Steve used it to make his sample Put_Image routines agnostic to which concrete stream implementation is used.

Erhard would prefer to drop FIFO. He notes that a “stream” is inherently a “FIFO”, it's a different way of saying the same thing. He suggests:

```
Ada.Streams.Storage
   Storage_Stream_Type
   Ada.Streams.Storage.Unbounded
      Stream_Type
   Ada.Streams.Storage.Bounded
     Stream_Type
```

After some inconclusive discussion, we take a Straw Poll. That leads to some discussion and eventually a number of people changing their initial votes. The final tally is:

FIFO/Bounded_FIFO (4: Jeff, Richard, Brad, Jean-Pierre);
Unbounded/Bounded (6: Steve, Erhard, Ed, Justin, Randy, Tucker)
Abstain (2: Bob, Gary).

This isn't important enough to get complete agreement. We'll go with the new suggestion.

Approve AI with changes: 9-0-2 Bob, Brad abstain

AI12-0330-1/00 Add items to the Glossary

Tucker half-heartedly volunteers to do this. No one else says a word. Tucker gets the AI.

Jeff will help as needed. He suggests Tucker look at what he already sent. [Editor's note: Jeff's mail is filed in the ! appendix of the AI.]

No vote was taken, as there is no proposal yet.