Minutes of Electronic ARG Meeting 60B

14 January 2019

Attendees: Steve Baird, John Barnes, Randy Brukardt, Gary Dismukes, Bob Duff, Jeff Cousins, Brad Moore, Erhard Ploedereder (starting at 12:45), Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega (starting at 11:05).

Observers: Edward Fish (starting at 12:45), Richard Wai.

Meeting Summary

The meeting convened on Monday, 14 January 2019 at 11:01 hours EST and adjourned at 14:05 hours EST. The meeting was held using Zoom. The meeting covered all of the regular AIs and a number of Amendment AIs on the agenda.

AI Summary

The following AIs were approved with editorial changes:

AI12-0213-1/03 Unify record syntax (12-0-0)
AI12-0236-1/04 declare expressions (8-1-3)
AI12-0242-1/08 Shorthand reduction expressions for objects (12-0-0)
AI12-0262-1/05 Map-Reduce attribute (9-0-2)
AI12-0299-1/01 The permission of D.2.1(10.1/2) allows too much (11-0-0)
AI12-0300-1/01 Annex G for Fixed * Integer (11-0-0)
AI12-0301-1/01 Predicates should work like constraints for types with Default_Value (11-0-0)

The intention of the following AI was approved but they require a rewrite:

AI12-0266-1/09 Parallel Container Iterators (7-0-4)

The following AI was discussed and assigned to an editor:

AI12-0208-1/07 Predefined Big Numbers support

The following AI was discussed and voted No Action:

AI12-0190-1/08 Anonymous functions (10-0-2)

Detailed Minutes

Welcome

Steve says “welcome” and continues by saying that completes the agenda item.

Apologies

Jean-Pierre Rosen said that he would be on a train and unable to attend.

Previous Meeting Minutes

No one has an changes or comments on the minutes. Approve minutes: 10-0-0.

RM Review Plan

Randy explains the plan for reviewing the draft Ada Standard before the next meeting. We will distribute a draft Standard with all of the proposed changes to date included outside of the containers changes in about 2 weeks. He will assign people approximately equivalent sections of the Standard to review. Hopefully, these can be turned around in time for our next meeting, so we can begin to address the issues that inevitably will come up.

Date and Venue of the Next Meeting

Randy proposes the next electronic meeting to be on February 11th at 11 AM EST. He asks if two weeks is enough review time before the next meeting. Tucker suggests that for deadline driven people such as himself, it is the deadline that matters more than exactly when it is. (He'll do it all on the due date no matter when that is.)

No one has any objection to the February 11th date, so that date is chosen.

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

Randy notes that we need to decide soon on dates and a location for the fall meeting. Steve asks Tucker if there is some equivalent to HILT. He says that there is not. Tucker suggests that a date near AdaCore's HUB week would work best for the AdaCore West Coast people, since they'll be on the east coast anyway. But the dates for that have not yet been set, either. Tucker offers the AdaCore Lexington office again. AdaCore's New York office is also offered. We'll defer this item until next time.

Unfinished Action Items

Steve Baird has the only unfinished action items, the “parts is parts” AI, the “nested generics and type invariants” AI, and the “dynamic accessibility is a pain to implement” AI. He will work first on “parts is parts”, he notes that there doesn't seem to be any alternative other than checking every use of a form of “part” for dynamic vs. static part.

Current Action Items

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

Everyone:

- Review their assigned portion of the draft Standard, after it is posted by Randy.

Steve Baird:

- AI12-0016-1
- AI12-0191-1
- AI12-0208-1
- AI12-0210-1

Randy Brukardt:

- Post a draft Standard as soon as practical, and assign review sections.

Editorial changes only:

- AI12-0213-1
- AI12-0236-1
- AI12-0242-1
- AI12-0262-1
- AI12-0299-1
- AI12-0300-1
- AI12-0301-1

Tucker Taft:

- AI12-0266-1
- Propose a chunk specification for the parallel reduce attribute (see discussion of AI12-0242-1).

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 16 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

AI12-0190-1/08 Anonymous functions

The syntax was changed to anonymous_function (rather than function_expression, which was confusing with expression function), and we changed the rules so that the parameter names are properly declared, which should fix the conformance issues. Randy notes that Tucker had illustrated how confusing the original naming was by having originally written expression_function when he meant function_expression, and then repeating this error in every draft he sent (presumably because it didn't get found by search-and-replace).

No other typos or technical issues are raised.

Randy asks if we want this. He argues that this feature is premature, hard to implement, and harms readability. It's impossible to put contracts on this feature, or on the anonymous access-to-subprogram that it is tied to. That harms static analysis.

Tucker asks for a vote, saying we've discussed all of this before.

Randy suddenly remembers that this has an implicit type of closure as well, since it can generated inside of a declare expression. We pretty much agreed we didn't want anything other than simple objects in a declare expression, and this definitely is not a simple object. It might cause functional people to start putting everything they can into a declare expression to get this closure-like behavior, which would be bad for readability, turning Ada expressions into LISP with a dozen closing parenthesis at the end of an expression.

We try the vote.

Approves AI with changes: 6-5-1.

Randy votes against, reasons given above. Gary votes against: sees it hurting readability, and not sufficient benefit for the cost.

Jeff and John vote against. Jeff agrees with Randy; he had another comment but can't find it. John doesn't like anonymous things in general.

Erhard also votes against. He does not like the mapping onto access. He is concerned about accessibility issues, especially for objects being returned. He notes it's not really a lambda, which is what he really wants. Tucker notes that full lambdas really only work for garbage-collected languages. He tried to devise a set of rules for objects that would be safe for a lambda, but he was unable to do so. He says Erhard is welcome to try.

Ed abstains.

This vote fails; with 12 voters, there can be no more than 2 against. (Not that anyone would want to argue that any proposal that nearly half of the group opposes should go in anyway.)

We then get bogged down on the next step. We can put the AI on Hold, or make it a No Action. The question really boils down to whether we would want to work on this exact version again. More people seem to suggest No Action

rather than Hold. Tucker notes that there isn't much practical difference, as we can always restart work on it using this version as a starting point, so long as it is not deleted from the disk. And we keep all of our AIs forever.

No Action: 10-0-2 Abstain: Justin, Tucker.

AI12-0208-1/07 Predefined Big Numbers support

Steve asks that we touch a moment on a question he had relating to the Big Number AI. It has been proposed to split the AI, putting the bounded numbers into a separate AI.

There are a number of design issues with the bounded form that haven't been clearly decided (especially, the proper form for describing the bound). No one objects to such a split. We then return to the regular agenda.

AI12-0213-1/03 Unify record syntax

Randy notes that the syntax in !proposal should be deleted, it's obsolete. And the word “record” is missing in the syntax in the !wording.

Erhard would like the !ASIS clause to not ask a question. It should be replaced by “ASIS needs to reflect the syntax change.”.

Erhard wonders why we would want a C-Test for this feature. Randy says that a C-Test would have a rather low priority, but there is always value to trying a new feature in a correct program. B-Tests don't test code generation or linking; it seems unlikely that there would be a problem there for this feature, but a lot of compiler bugs occur where one least expects them.

In the !proposal, “thusly” doesn't make much sense. Modify it as follows:

> We propose introducing an "end record identifier" syntax for record definitions [(]and [thusly] representation clauses[)] so that their syntax more closely matches program units such as subprogram_declarations and package_declarations. See the code {fragments}[segments] below for an overview of the proposed change:

Gary: !summary should mention representation clauses.

Approve AI with changes: 12-0-0.

AI12-0236-1/04 declare expressions

The type invariant example is missing the **declare** keyword.

There probably ought to be an example in the RM text. We need to use existing declarations, so someone (that is, the editor, Randy) needs to create one.

Erhard notes that there is a semicolon missing in the example starting X : T ...

Someone would prefer not to see **begin** in the syntax. We do not want to discuss the syntax any more, it was difficult to reach a consensus, and it is unlikely that more discussion would change the conclusions.

Someone is concerned that dumb tools (such as editors) that support Ada could be confused by the lack of an **end** after **begin**. Tucker notes that such tools don't work on Ada 2012 because of **if** and **case** expressions, which are missing the **end if** and **end case**. Ada source tools have to be aware of parentheses, or they will be horribly confused. This feature doesn't change that fact, and even dropping **begin** wouldn't change anything for this feature (because **declare** expects a **begin** and **end**).

The examples look weird with the **begin** at the end of the line. Tucker thinks that the later example looks fine. Move the **begin** to the beginning of the line in the examples.

Randy asks if we want to allow **others** in aggregates. Tucker says that if conditional expressions allow that (and they do), declare expressions should also allow **others**. Otherwise, adding a declare expression to simplify an existing conditional expression could fail to work, which would be obnoxious. Bob says that the wording needed to support **others** is in the !discussion part, move that to the !wording section.

Tucker and Bob note this is the number #1 priority of the SPARK people.

Gary notes that “post-condition” should not have a hyphen in the !problem.

The semicolon at the end of the Dynamic_Predicate example should be dropped, in order to be consistent with the other two examples.

Approve AI with changes: 8-1-3.

Jeff votes against. He thinks that people will write much more complex postconditions and essentially repeat bodies in the postcondition.

Abstain: Erhard, Gary, John.

AI12-0242-1/08 Shorthand reduction expressions for objects

Ed: Last line of proposal: “... parellel keyword specified.”

Do we need this? Ed says yes, it seems likely. Randy notes that manual chunking using a full chunk_specification leads to using this to create a final result.

Tucker asks whether reduce attributes should have chunk support? We could do something like that in AI12-0262-1:

> [**parallel** [(expression)]] iterated_component_association

Seems like a useful idea. But, let's do that separately. Tucker takes an action item to make a proposal in this area.

We discuss how that would get added to the shorthand. Randy makes the point that a shorthand is just that, it's easy enough to write out the value sequence if you need to specify a number of chunks. So we won't include chunk specifications in the shorthand.

Gary notes that in the !problem two semicolons (;) should be colons (:). [Editor's note: There's actually three such semicolons.] “eg. The” should be “For example, the...”.

Gary wants a comma in the !proposal “... AI12-0262-1{,} which ...”. We're all surprised (not!). In the next paragraph: “ synatically “, “syntatically” (two uses, both misspelled).

John: The prefix of the attribute is “O”, which is confusible with '0'. The Standard usually uses X as the prefix for objects, do that here.

[Editor's note: The description of Parallel_Reduce is missing two words “...a result equivalent to {replacing the} prefix of the...”]

Approve AI with changes: 12-0-0.

AI12-0262-1/05 Map-Reduce attribute

Drop the production

> reduction_expression ::= reduction_attribute_reference

from the syntax, as this isn't used anywhere.

Tucker notes that we need to make the connection between reduction expression and the attribute in the introduction:

> A reduction expression is represented as an attribute_reference using the Reduce attribute.

Ed note a spelling error in the third paragraph of the Legality Rules: “reservered word **others**”.

Jeff thinks “first” is out of place in the Dynamic Semantics. Replace that by:

> For a reduction_attribute_reference, the value_sequence is evaluated first.

Tucker: In the first sentence of Static Semantics, “Otherwise” is weird.

> The nominal subtype of the index parameter of a value_sequence is that of the discrete_choice. The nominal subtype of the loop parameter of a value_sequence is as defined for an iterator_specification (see 5.5.2).

In the attribute definition, “Yields the final result of the iteration.” should be “The result of V'Reduce is the final result of the iteration.”

Similarly, “completed, {V'Reduce} yields the result of the last call to Combiner.”

First sentence of Dynamic Semantics, Steve would want:

> The reduction_attribute_designator is then evaluated, and then the reduction is evaluated.

Syntax:

> *reduction*_identifier

Single quotes around the square brackets in the syntax.

Ed comments that the !discussion uses “Reduction” as the name of the attribute in the My_String example.

Gary would like to turn around the last paragraph of V'Reduce:

> If the value_sequence of the reduction_attribute reference produces an empty sequence, V'Reduce yields the value of the Initial_Value expression.

Approve AI with changes: 9-0-2 (Abstain: Bob, Gary)

AI12-0266-1/09 Parallel Container Iterators

Brad explains the changes. The interface is now more like the other iterators.

Tucker: “First {(} Object” is missing a paren.

We should drop all of the Nonblocking aspects here, because that just restricts use of the interface arbitrarily. Tucker notes that it doesn't make sense to tie whether the interface is nonblocking to the Cursor type and Has_Element function. Randy complains that the containers would need an explicit requirement for their interfaces to be nonblocking. The group thinks that would be a better model. Drop the overridings as well, they won't be needed.

[Editor's Note: That Nonblocking requirement needs to go in ***this*** AI, along with the change in the type name for the container's iterators. We could probably put that into a blanket rule in A.18 as an Implementation Requirement. Something like:

> If an instance of an Ada.Containers generic package is nonblocking, then the specific type of the object returned from a function that returns an object of an iterator interface, as well as the primitive operations of that specific type, shall be nonblocking.

End Editor's Note.]

Brad also added chunk_specification syntax and semantics to this AI.

The AI includes changes to ensure that all of the language-defined containers provide parallel iterators.

Richard would prefer to use Has_Element_in_Chunk and Next instead of Has_Element and Next_in_Chunk.

Randy notes that you can't use both existing routines in a chunk of a parallel loop, so there doesn't seem to be any savings; Richard's change just replaces one new routine by a different new routine.

Tucker suggests that it is safer as currently defined, since you don't get some other chunk's pointers from the interface. Hoping users use the right Has_Element to test is iffy. (Of course, this interface will mainly be used by compilers, so this isn't that compelling.)

Drop Last_in_Chunk and Previous_in_Chunk, as those aren't used by any proposed wording. Someone asks why we need this interface in that case. Because the Parallel_Reversible interface provides both interfaces, but no new operations. You can't have both at the same time, it is Parallel and Reversible iterator.

Drop "_in_Chunk" consistently. Overloading works! Next with a chunk index is just fine.

Tucker suggests putting Parallel_Reversible_Iterator next to Parallel_Iterator. That doesn't work, because it has to follow the operations. He quickly withdraws this suggestion.

Should this interface be renamed? Parallel_and_Reversible_Iterator? Parallel_or_Reversible_Iterator? Parallel_with_Reversible_Iterator? There's no consensus.

Tucker suggests adding an AARM Note on this topic somewhere. Perhaps something like:

> AARM Note: It's not possible to use "reverse" with "parallel"; this interface is intended to allow an iterator object to support both "reverse" sequential loops and "parallel" loops (it also will support forward sequential loops).

Tucker suggests improving the English description of this interface: 5.5.1(6):

> A parallel reversible iterator type is a type descended from the Parallel_Reversible_Iterator interface from some instance of Ada.Iterator_Interfaces{, which supports either parallel or reversible iteration.}

In the definitions of 5.5.1(6) and 5.5.1(11), we need to say that an object of a parallel reversible iterator is both kinds of iterator.

> An object of a parallel reversible iterator type is both a parallel iterator object and a reversible iterator object.
> An object of a parallel reversible iterable type is both a parallel iterable object and a reversible iterable object.

Randy would like Tucker to redraft the Dynamic Semantics, because Brad's redraft is missing the object creation and evaluations. And it's not obvious how to add them. Tucker left out object creation from the regular loop description, but that is OK as creating a discrete object can't have a side effect. Since type Cursor can be any kind of type, it could have a non-trivial default initialization or finalization.

So Tucker will take this AI to do wordsmithing. He asks Randy to give the details of what missing, Randy says he sent that in e-mail last night. [Editor's note: The message in question was sent to the ARG list Sunday, January 13th at 12:27 AM CST. Titled "Re: [ARG] AI12-0266-1 Parallel Container Iteration"]

Approve intent: 7-0-4. Abstain: Bob, Ed, Gary, Justin.

AI12-0299-1/01 The permission of D.2.1(10.1/2) allows too much

Bob suggests combining the sentences of D.2.1(7/5):

> A call of Yield and a delay_statement are task dispatching points for all language-defined policies.

Delete the AARM note following this.

Approve AI with changes: 11-0-0.

AI12-0300-1/01 Annex G for Fixed * Integer

A vendor submitted this after they had a customer who was confused by the original wording.

Fix the summary: "For fixed times integer multiplications, the integer type is Standard.Integer."

Gary would like to replace “vice-versa” with “vice versa”. There are 6 with the hyphen, and 4 with the space. The correct spelling, according to John, is “vice versa”. Let the editor decide (put into AI12-0066-1 if not done).

Approve AI with changes: 11-0-0.

AI12-0301-1/01 Predicates should work like constraints for types with Default_Value

Bob says that he thinks this is the correct rule.

Bob: “inplicit” in the last line of the question.

Approve AI with change: 11-0-0.