Minutes of Electronic ARG Meeting 60E

11 March 2019

Attendees: Steve Baird, Randy Brukardt, Gary Dismukes, Bob Duff, Brad Moore (arrived at 11:10), Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega.

Observers: Richard Wai.

Meeting Summary

The meeting convened on Monday, 11 March 2019 at 11:01 hours EST and adjourned at 14:01 hours EST. The meeting was held using Zoom. The meeting covered some of the AIs on the agenda.

AI Summary

The following AIs were approved:

AI12-0250-1/03 Iterator filters (9-0-1)
AI12-0282-1/03 Atomic, Volatile, and Independent generic formal types (10-0-0)
AI12-0319-1/02 Nonblocking for Unchecked_Deallocation is wrong (6-0-4)

The following AIs were approved with editorial changes:

AI12-0234-1/05 Compare-and-swap for atomic objects (8-0-2)
AI12-0317-1/08 Simplifying the rules for newly constructed objects (10-0-0)
AI12-0320-1/02 Changes from the RM review (10-0-0)
AI12-0321-1/02 Support for Arithmetic Atomic Operations and Test and Set (9-0-1)
AI12-0322-1/03 Equivalence for the target name symbol (10-0-0)
AI12-0323-1/01 Implementation Advice for the CPU aspect for protected types (8-0-2)

The following AI was discussed and assigned to an editor:

AI12-0240-5/02 Heap object ownership for abstract data types

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

Erhard Ploedereder said that he is on vacation and probably won't be able to attend. Jeff Cousins said that he is on “holiday” and won't be able to attend. Brad Moore joined 10 minutes late because he was confused by Daylight Savings Time having started on Sunday.

Previous Meeting Minutes

Randy notes that various people had sent corrections to the minutes, these all have been handled in the posted minutes. No one has any further changes on the minutes. Approve minutes: 9-0-0.

Date and Venue of the Next Meeting

There had been an e-mail discussion that appeared to settle on April 9th for our next electronic meeting. After a bit of discussion, we agree to the April 9th date, 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.

RM Review Plan

Randy thanks everyone for getting their reviews in on time. He managed to process 5 of the reviews before this meeting and skimmed most of the others. [Tucker still had half of his to do, which he completed a few days after the meeting. 5 lashes with a wet noodle. - Editor.]

There will be a second round of review that is simultaneous with the WG 9 review period.

Current Action Items

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

Steve Baird:

- AI12-0016-1
- AI12-0240-5 (create additional examples of the use of this feature)

Randy Brukardt:

Editorial changes only:

- AI12-0234-1
- AI12-0317-1
- AI12-0320-1
- AI12-0321-1
- AI12-0322-1
- AI12-0323-1

Justin Squirek:

- AI12-0240-5 (report on implementation effort for this proposal for GNAT)

Tucker Taft:

- AI12-0240-5

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 18 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

AI12-0234-1/05 Compare-and-swap for atomic objects

Brad describes the changes.

“Ada.” should be “System.” in C.6.3 “Static Semantics”. [Editor's note: Actually, the entire paragraph is old junk which needs to be deleted.]

Randy suggests merging C.6.1 and C.6.2.

Steve notes that it should say “object{s}” in that introductory paragraph.

“...for creating lock-free data types.” should be “for supporting lock-free synchronization”.

The introductory sentence in C.6.2 should be combined and/or deleted. We'll leave that to the editor.

“... processor of {the} logical thread of control ...”.

Richard asks if the “Is” prefix makes sense for the subprogram Is_Lock_Free. It isn't dependent on the value of the parameter. Tucker notes that “Is_Terminated” and the like are defined.

Steve wonders why the parameter is necessary. It's about the alignment (or possibly the address). Richard wonders if the instance should be disallowed in cases where it could not be lock-free. Tucker notes that would be hard to do, we've tried hard to avoid “magic” rules on generic instantations. Also, Tucker suggests that it still would be useful for portable access to these operations, even if not lock-free (not everyone cares about that property).

Upper-case 'K' in Is_LocK_Free.

Gary suggests “spinlock” should be “spin lock”.

Tucker notes `end {System.}Atomic_Operations.Exchange;`

“*-- obtain the [spin]lock*”.

“the value of Value”, “the value of Desired”. (Values aren't denoted). “written into” should be “assigned”.

We discuss whether it makes sense for these to be functions; they seem more like procedures with **out** parameters. Tucker notes that these are usually used in a loop, and it would be a pain to use a procedure that way (needing to declare local objects).

Steve hates the name “Expected”, since that is only one of the two uses. It doesn't describe the returned value at all. Tucker says that Wikipedia uses “Old” and “New”.

Jean-Pierre notes that in the example, we don't need the conversion of “True”.

Tucker suggests replacing “Expected” by “Prior”. That is what the result is in either case. We agree to that change.

Approve AI with changes: 8-0-2 (Abstain: Steve, Gary).

AI12-0240-5/02 Heap object ownership for abstract data types

The problem is, when using Global, there is no way to build data structures using access types without involving the world. One has to mention either the memory used by the storage pool, the access type (which often is private), access to the designated type (which also is often private), or give up and use **all**. This happens because a dereference has to include any object that it *could* reference, which is the entire universe of objects of the right type (if the access designated type form is used) or in the right pool (for pool-specific types, using the access type form).

This is a problem when writing the containers in particular, since we need to use Global => **null** in many places to allow parallel execution even when the checking is turned on (which is the default for parallel operations).

Tucker gives an overview of ownership proposal, using a few slides that he'd prepared beforehand.

Ed worries that we will end of up with another coextension or accessibility rules. He fears that we will find new rules that need to be added repeatedly. Tucker notes that the current version is not trying to make the checking perfect, so

finding of new “holes” in the checking may not require any fixes. In this way, it is more like type invariants than coextensions.

Ed also worries about the implementation cost. Randy says that he didn't see anything particularly hard in his review, but of course conclusions about one implementation are not necessarily useful for another. Tucker says that we could include an option to eliminate the runtime cost. Randy notes that the runtime checks are normal runtime checks, and presumably they would have an associated check name and could be suppressed. The AI probably should include that possibility so it doesn't get overlooked.

It would be very useful to get a read on implementation cost, especially for GNAT. We then discuss in general terms how AdaCore will implement Ada 2020.

Eventually, Justin volunteers to look at the effort needed to implement these rules in GNAT. He is given an action item to do that. Steve volunteers to work out additional examples of the use of these features. He is given that as an action item. Tucker suggests that Steve look at the implementation of a multiway tree, which has two back pointers (one for the child lists, which are doubly linked, and one pointing at the parent node). Steve wonders what multiway trees we are referring to; he is directed to A.18.10 in the RM. Steve's excuse is that he doesn't read the back of the reference manual that often.

We did not take any kind of vote after this discussion.

AI12-0250-1/03 Iterator filters

Tucker describes the changes. Mostly, 4.5.10 was rewritten to use iterated_element_association. It is a better match when all of the semantics are explicit.

Tucker wonders whether we should disallow **reverse**. We discuss this without much of a conclusion. It seems odd for most uses, but it could be valuable for non-commutative operations. Randy suggests that it should be illegal in any case when **parallel**, as we don't allow that in any other parallel case. Tucker says that unlike other operations, reduction is defined such that whether or not **parallel** is used that the result is the same. We decide to leave the rule as it is.

Approve AI: 9-0-1 (Randy abstains)

AI12-0282-1/03 Atomic, Volatile, and Independent generic formal types

Tucker describes the changes. He was able to simplify most of the rules associated with generic formals by just saying that an atomic type requires an atomic formal type (and vice-versa). There isn't such a rule for Independent (an independent type can be passed to a non-independent formal).

Steve asks about whether atomic discriminants are disallowed. There doesn't seem to be any such rule, but he doesn't explain the problem (if any), and in any case, this question doesn't have anything to do with atomic formal types. If there is a problem, he should explain it and propose a solution in a future AI.

Approve AI: 10-0-0.

AI12-0317-1/08 Simplifying the rules for newly constructed objects

We discuss the critical problem that put this AI back on the agenda. “Constituent” already has a definition in Ada. Luckily, it is similar, but broader than the new term. We eventually settled on “operative constituent” as the new term.

Fix spelling in the To Be Honest note: “opeerative”.

Every name or expression [comprises]{consists of} one or more ...

Brad: in the summary, “construct like ...” is weird. Does it need a hyphen? No. We eventually decide that there are missing “a”s:

> ... a construct like {a} parenthesized expression or {a} conditional expression ...

Approve AI with changes: 10-0-0.

AI12-0319-1/02 Nonblocking for Unchecked_Deallocation is wrong

Randy tries to explain the changes. We needed to extend the Nonblocking description to be able to describe the blocking status of a storage pool (since Unchecked_Deallocation is going to call Deallocate; and the same issue would arise for any generic that tried to call an allocator of a generic formal type). He also discovered that the rules which are needed to describe the initialization and finalization blocking of a type were missing. These are used extensively in the containers contracts, as well as in Unchecked_Deallocation, so this has to be done correctly. Finally, there are rules to specify Nonblocking for the Standard pool(s), and also for the implementation-defined pool used when Storage_Size is specified.

Tucker asks about classwide operations. These are whatever the specific type's Nonblocking is specified to be. There are rules which prevent derived types and subprograms from differing, unless they change from allowing blocking to nonblocking. This means that any routine that can be dispatched to has the same or stronger Nonblocking. The generic body assume-the-worst rules apply to such types as well.

Steve wonders if concurrent access to the Standard storage pool is covered here. No, but that's always been assumed. He says it should be explicitly stated. Perhaps, but that's not what this AI is about; he is welcome to propose an AI with some wording to do that.

Approve AI: 6-0-4 (Gary, Ed, Justin, J-P abstain)

AI12-0320-1/02 Changes from the RM review

Randy goes through each of the each the issues.

The wording of the documentation requirement is still weird. Change the inserted “if” to a “when”.

Approve AI with changes: 10-0-0.

AI12-0321-1/02 Support for Arithmetic Atomic Operations and Test and Set

Richard complains about the “and” and “or” for signed types. We don't want those on signed type. Perhaps they should be in a modular package, but that's a different AI. We mostly want add and subtract, since these are clearly useful and many machines have native instructions for doing these atomically. (The widely used Intel x86 processors do, for instance.)

Steve wonders what Atomic_Test_and_Set does if the value contains 37 and the implementation-defined value is 1. He notes that the type is modular, so a user could assign any value they like to it. We try to figure out what the C standard says in this case, but it proves inscrutable. We think that C assumes any nonzero value is True, that's probably what we should do here.

We wordsmith for a while, ending up with:

> Atomic_Test_And_Set performs an atomic test-and-set operation on Item. Item is set to some implementation-defined nonzero value. The function returns True if the previous contents were nonzero, and returns False otherwise.

Since there are only five subprograms in Arithmetic, we should just describe each individually. Which will get rid of the “i.e.” that we're not allowed to have anyway.

Approve AI with changes: 9-0-1 (Gary abstains)

AI12-0322-1/03 Equivalence for the target name symbol

Steve asks if aliased passes through the target symbol? Tucker is confused. Steve explains that he is wondering whether the target name symbol is aliased if the target object is aliased. One would expect that, it could be used even though the view is constant if the usage is for an access-to-constant type.

Randy checks 3.10, and the answer appears to be no. There are a bunch of rules about specific declarations, and then about renamings of those declarations. Tucker suggests adding a "other properties are the same" as with renames in 8.5.1.

Tucker will take this off-line to reword this.

Steve notes that this is a better model for this AI, than trying to make any equivalence fit – which is necessarily going to be an imperfect fit.

We turn to discussing AI12-0323-1 while Tucker crafts wording.

Afterward, we return to this AI.

> ... evaluation of each target_name yielding a constant view of the target whose properties are otherwise identical to those of the view provided by V.

Delete the change to 5.5.1(3/5). Change the AARM note to refer to the Dynamic Semantics.

Fix the ASIS section to say no change is needed (any changes should be in AI12-0125-3).

Approve AI with changes: 10-0-0.

AI12-0323-1/01 Implementation Advice for the CPU aspect for protected types

Put location of the change into the !wording section.

Approve AI with changes: 8-0-2 (Gary, Steve abstain)