A New Profile Based on Ravenscar

Pat Rogers
J. Ruiz, T. Gingold, P. Bernardi
[rogers|ruiz|gingold|bernardi]@adacore.com

Ada Europe 2017
Vienna, Austria
The GNAT Extended Ravenscar Profile

• A new profile based on Ravenscar

• For use in a subset of the applications intended for Ravenscar

• Specifically, for real-time and embedded applications *not* requiring most rigorous forms of analysis
  – For example, certification or safety analysis

• But do require schedulability analysis
  – Unless simply embedded
Does Not “Replace” Ravenscar

• Use Ravenscar when maximum simplicity is required
  – RTL is less complex, less expensive to analyze

• Use Ravenscar when maximum efficiency is required

• AdaCore is shipping both profiles and will continue to do so
  – Ravenscar profile in the ravenscar-sfp-* runtimes
    ▪ For certification
  – Extended profile in the ravenscar-full-* runtimes
Why Another Profile?

- Ravenscar is necessarily restrictive
  - For sake of certification and safety analyses
  - For sake of maximum efficiency
  - For sake of easiest schedulability analysis

- A loss of expressive power inevitably results from a restrictive subset

- This loss can be mitigated when only predictability and relative efficiency are required
What Does the New Profile Add/Allow?

- **Multiple protected entries per PO**
- **Multiple queued callers per protected entry**
  - Entry queue depth can be greater than 1
- **Somewhat relaxed entry barriers**
  - Via new restriction “Pure_Barriers”
- **Relative delay statements**
  - E.g., to protect electro-mechanical relay burnout
- **Use of Ada.Calendar**
  - E.g., for time-stamping
The New Pure_Barriers Restriction

- Applied instead of Simple_Barriers
- Allows more expressive entry barriers
- Addresses implementation freedom regarding number of times a barrier expression is evaluated
- Therefore, barrier content remains restricted
- No side-effects and no exceptions possible
- No recursion either
Constructs Allowed by Pure_Barriers

- Variables local to the protected object (private part)
- Discriminants for the protected object
- Numeric literals
- Enumeration (and hence character) literals
- Named numbers
- Relational operators
- Logical operators (and, or, xor)
- Short-circuit control forms (and then, or else)
- The logical negation operator (not)
- The Count attribute for entries
protected body Barrier is

entry Wait when (Wait'Count = Capacity) or Release_Others is
begin
    Release_Others := Wait'Count > 0;
end Wait;

function Value return Positive is
begin
    return Capacity - Wait'Count;
end Value;

end Barrier;
protected body Bounded_Buffer is

entry Put (Item : in Element) when Count /= Capacity is
begin
  Values (Next_In) := Item;
  Next_In := (Next_In mod Capacity) + 1;
  Count := Count + 1;
end Put;

entry Get (Item : out Element) when Count > 0 is
begin
  Item := Values (Next_Out);
  Next_Out := (Next_Out mod Capacity) + 1;
  Count := Count - 1;
end Get;

...

end Bounded_Buffer;
What is the Cost?
Canonical Protected Action Semantics

- Recall “writers” are protected procedures and entries; they can change a PO’s state

- Whenever a writer completes, all entries are evaluated and one with a True barrier and a queued caller will execute, if any

- Each entry is a writer, so completion triggers another iteration of evaluation, selection, and possible execution

- Iteration repeats until no more entries can be executed
In Ravenscar, No Iteration Involved

- A protected procedure can trigger a protected action but there would be at most one entry to execute

- That entry could have at most one caller

- Thus run-time library routine is very simple
  - No loop
  - No queue processing
  - No repeated barrier evaluations
So What Is the Cost?

- Increased execution time for protected procedure and entry calls
- Increased execution time to call some attributes
- Blocking term can be increased
- All due to iterative protected actions
- Note a PO with no entries is not affected
  - Specialized RTL routine called
How Much Overhead?

- Measured using the ESA Ravenscar Benchmarks (ERB)

- Comparing same Ravenscar benchmarks on RTLs providing Ravenscar and new profile
  - Only difference is the profiles

- Thus measuring overhead of new profile

- Expressed in terms of percentages relative to Ravenscar
• Keep in mind limitations of percentages
  – From 2 instructions to 3 would be 50% increase

• Entry call, barrier open: 54% slower
  – Due to iterative protected action semantics

• Entry call, barrier closed: 25% slower
  – Caller is queued

• Protected procedure call, when PO contains a closed entry: 13% slower
  – Check for other open entries but nothing else
Constant Overhead Percentages (2 of 2)

• Call to 'Count for an entry, made from a protected procedure within that same protected object: 21% slower
  – Value must be computed

• Call to 'Caller in an entry body: 58% slower
  – “Last task in” means caller may not be executor

• Actual times are still small and the implementation is still simple

• Conclusion: yes, there is overhead but acceptable
What About Schedulability Analysis?

• **We use the “last task in” implementation for protected actions**
  - The last task in the PO evaluates all entry states and executes entry bodies of all open entries on behalf of queued callers
  - Avoids task switch for each entry body execution

• **Affects the “blocking” term in the analysis**
  - Time a task is blocked by lower priority tasks
  - Bounded and quantifiable

• **Thus analysis remains possible**
The Blocking Term Value

• “Last task in” means calling one entry may require time to execute all entries in that PO, for all queued callers

• Worst case number of callers is all other tasks

• But for each entry, we can specify max callers via new aspect Max_Queue_Length

• Worst case blocking term for any one entry is the sum of times to execute all entries in that PO, for max callers per entry

• Thus blocking bound is reduced to a user-controlled value, depending on design
Summary

- A profile complementary to Ravenscar
  - When most stringent analyses not required

- Provides significant expressive power gain
  - Many protected object idioms now allowed
  - No need for “delay until Clock + Interval” idiom

- Predictability and efficiency retained

- Schedulability analysis remains possible

- We hope to have it in Ada 2020
  - With a much better name