Rigorous Software Development
Second Event-B Exercise sheet

Deadline: Dec. 20th 2019, 20:59

Manuel Carro
manuel.carro@upm.es

December 7th, 2019

1 Introduction

The goal of this exercise is to formally prove the correctness of an algorithm to perform binary search in a sorted array. We want to know the position r of a value v which we **know** is in a vector f that stores values in non-decreasing order (Section A). We previously showed and proved correct (w.r.t. some invariants) an Event-B specification which performs this search using (i) a random choice in the vector (Section B) and (ii) a random choice in a shrinking window bounded by p and q (Section C).

2 Tasks in this exercise

We want to eliminate the non-determinism in the selection of r: we want r to be placed in the point in the middle of p and q (approximately; when $q - p$ is odd the midpoint does not correspond to a position, but that should not be a problem). The rest is as before: either p or q is updated depending on where $f(r)$ lies w.r.t. v.

Your tasks are:

1. Download and install RODIN.[1] Remember to install the *Atelier B* provers. See

[1] I suppose you have already done it!

http://babel.ls.fi.upm.es/rsd/#rodin-tool

in the course web page and the instructions at

http://babel.ls.fi.upm.es/teaching/rsd/Slides/03-binary-search-refinement.pdf

In addition, the page

https://www3.hhu.de/stups/handbook/rodin/current/html/proving_perspective.html

has a good explanation of the meaning of every item in the proving perspective — that should cover what we saw in the lecture on the refinement of the binary search. I recommend reading it thoroughly.

2. Download the `.zip` file with the project up to the second version (there is a link in the course web page) and import it into Rodin. See

   https://www3.hhu.de/stups/handbook/rodin/current/html/sect0033.html

   and the slides for instructions on how to import a project.

3. Generate your version of the second refinement with the selection policy mentioned above. It is possible to refine either `BS_M0` or `BS_M1`. Refining `BS_M1` should leave fewer proofs to discharge, so that is my recommendation.[2]

   There are two ways to do that:

   - (Recommended) Have Rodin to generate a template for you and change it:
     (a) Right click on `BS_M1` (or `BS_M1`), select Refine, give the new machine a name. The refined machine will be duplicated and all guards and actions will appear dimmed.
     (b) If you need to edit any event, click on the extended keyword so that it changes to not extended and you will be able to modify its code.

     Initially, you will not see invariants or invariant-related proof obligations in this refined machine — it is the same as previous one, and the invariants from the previous machine still hold. You can add invariants if you think they are necessary. Also, if you introduce new variables, you will have to add new invariants for them.

[2] The guards can remain the same, so there is nothing to prove in that respect. The actions should change, so a SIM(ulation) proof is necessary. Note the beauty that if the SIMulation of the actions in `BS_M2` is proved, then the invariants in `BS_M1` are preserved by the actions in `BS_M2`, because these can traverse only a subset of the states that `BS_M1` does!

- (Equally valid, basically same results, slightly more work) Create a new machine, declare that it refines `BS_M1` adding a `REFINES` section (see machine `BS_M1`), write the events that refine the previous events (you can use the same names) and declare what event refines every new event.

4. Prove in Rodin the proof obligations not automatically discharged (those marked in brown). I recommend repeating what I did in the binary search lecture to get used to the interface of Rodin (which amounts to lassoing, instantiating, and using P0). Reversing the implication inside the universal quantifier appear helps, but instantiating the correct part of left hand side is the key point.

   Save the project after every discharged proof to update the proof status.

5. When all the proofs are discharged, export the project to a `.zip` file. See

   https://www3.hhu.de/stups/handbook/rodin/current/html/sect0032.html

   and the instructions in the slides for the 04/12/2019 session.

6. Check that you have exported it correctly:

   - Temporarily rename the project you were working on.
   - Import the project you just saved as explained before (follow

     https://www3.hhu.de/stups/handbook/rodin/current/html/sect0033.html)

     or the instructions in the slides for the 04/12/2019 session..
   - Expand machines, check that all POs have been discharged.

7. Send it to me by email: manuel.carro@upm.es.

8. **Deadline: Dec. 20th 2019, 20:59.**

A Context: Axioms and constants

CONTEXT BS_C0

CONSTANTS

n

f

v

AXIOMS

axm1: $n \in \mathbb{N}_1$

axm3: $f \in 1 \,..\, n \to \mathbb{N}$

axm4: $\forall i \cdot \forall j \cdot (i \in 1 \,..\, n \land j \in 1 \,..\, n \land i \leq j) \Rightarrow f(i) \leq f(j)$

axm5: $v \in ran(f)$

END

B Random selection of a location within the vector

MACHINE BS_M0
SEES BS_C0
VARIABLES

r

INVARIANTS

`inv1`: $r \in dom(f)$

EVENTS
Initialisation

begin

`act1`: $r :\in 1 \,..\, n$

Should be dom(f) but that will force us to use PP to prove FIS. For simplicity we leave it like this.

end

Event final ⟨ordinary⟩ $\widehat{=}$

when

`grd2`: $f(r) = v$

then

skip

end

Event progress ⟨anticipated⟩ $\widehat{=}$

when

`grd1`: $f(r) \neq v$

then

`act1`: $r :\in dom(f)$

end

END

C Random selection of location place within a shrinking window

MACHINE BS_M1
REFINES BS_M0
SEES BS_C0
VARIABLES

r

p

q

INVARIANTS

inv1: $p \in 1 \,..\, n$

inv2: $q \in 1 \,..\, n$

inv3: $r \in p \,..\, q$

inv4: $v \in f[p \,..\, q]$

VARIANT

$q - p$

EVENTS

Initialisation

begin

act1: $p := 1$

act2: $q := n$

act3: $r :\in 1 \,..\, n$

end

Event final ⟨ordinary⟩ $\widehat{=}$

refines final

when

grd2: $f(r) = v$

then

skip

end

Event inc ⟨convergent⟩ $\widehat{=}$

refines progress

when

grd1: $f(r) < v$

then

act2: $p := r + 1$

act3: $r :\in r + 1 \,..\, q$

end

Event dec ⟨convergent⟩ $\widehat{=}$

refines progress

when

grd1: $f(r) > v$

then

act1: $q := r - 1$

act2: $r :\in p \,..\, r - 1$

end

END