Correctness by Construction
Second Event-B Homework

Deadline: Tuesday, March 26th 2024, 23:59

Manuel Carro
manuel.carro@upm.es

March 2nd, 2024

General Remarks

- This homework is individual.
- Please make sure you have read the course policy.
- Please turn in the solution to this homework no later than **Tuesday, March 26th 2024, 23:59**.
- If you experience problems with the assignment, please let me know as soon as possible. It may not be possible to implement last-minute changes / adaptations.
- The solution for this homework is an Event-B model developed using Rodin, which you should send me (see Section 4). I do not require any additional document, but if you think that submitting additional documentation / comments will help me understand your work, please feel free to do so. In such a case:
  - Preferably send me a PDF file.
  - Alternatively, send me a **good** scan of a (handwritten) solution, in PDF format. Please make sure that it is readable and, if you scanned the page, that it is not too dark, as this makes reading solutions difficult.
  - **Please do not send me Word, LibreOffice, Pages, etc. documents**.
    They may not be reproduced faithfully in all systems.
  - Please make sure to include your name in any document you send!
- In the models you submit, please make sure to include comments explaining the intended meaning of its parts: variables, invariants, guards, actions, etc. That will help me understand it.

1 Goal

The objective of this homework is for you to become acquainted with Rodin, a tool that uses interactive theorem proving to assist in the process of designing correct software, and to apply it to continue the development of an example that was presented in the classroom. We will (i) finish (or redo, depending on how far you could go during the lectures) the pending proofs for the model of the search in a sorted array and make it deterministic, and (ii) produce versions for the case when the element sought for may not be in the array.

I recommend to read thoroughly the Proving section of the website, as it contains practical tips that can be useful to discharge the proof obligations.

2 Setup

1. You should have Rodin and the Atelier B plugins installed. Please see the instructions at

   `https://wp.software.imdea.org/cbc/rodin-installation-and-tips/`

2. Import the model for this homework.

3. Rename the project:

   - Right-click on its name.
   - Select the menu entry "Rename".
   - Substitute "INITIALS" for your name's initials.

   I.e., for me it would be search-2024-hw2-MCL. Make sure that you use the same naming convention when you export the model to send it back to me once the homework is finished. Renaming models like this helps me identify the author of every model and avoid mistakes.

3 Tasks

3.1 Finish Undischarged Proofs

Your first task will be to discharge the pending proofs in the model search_bin_m2. A possible strategy to discharge the remaining proofs in this particular example is presented in the slides and was done in the classroom. The pending proof obligations in this machine can be discharged following similar steps.

3.2 Remove Non-Determinism

Non-determinism is a powerful modeling and specification tool, but it is not practical to generate efficient and predictable computing systems. Therefore, we want to remove the non-determinism in the selection of r: instead of choosing $r :\in p..q$ we want to set r to be in the midpoint of $p..q$. The rest does not need to change.

1. **Refine** the machine search_bin_m2 into search_bin_m3 (right-click on machine the name in the left panel, select Refine). Note that if you do not do a refinement, there will not be simulation / guard strengthening POs and the correctness of the refinement cannot be proven. Make the events you need to change non extended. Update the assignments to r actions.

   Which proofs are automatically discharged or not depends on how you change the model. You will likely have two WD proofs in green (proved) and two SIM proofs in brown (pending). These SIM proofs mean that it should be proven that the midpoint calculated for r has to be within the range $p..q$ so that the possible states of search_bin_m3 will be a subset of those of search_bin_m2. Therefore, the invariants in search_bin_m2 should hold in search_m3 as well.

   Depending on the changes that you make on the refined machine, GRD proofs may appear as well.

2. **Discharge the unproven POs**. A strategy similar to that used in search_bin_m2 should work: lasso, reverse implication, instantiate one of the applications of f in the LHS of the implication, invoke P0. Depending on your particular model, the proofs might be more complex and need further assistance.

3.3 v May Not Be in f

We worked so far with the assumption that $v \in \mathrm{ran}(f)$. In this section we will make separate models for the search process in a situation where v may not be in the array. For simplification, let us use the same project (search-2024-hw2-XYZ) we used in the previous tasks.

1. Create a machine (e.g., search_lin_maybe_v_m1) similar to search_lin_m1, but seeing **only** the context search_basic_c0, which does not include $v \in \mathrm{ran}(f)$. This machine should model a linear search in an unsorted array which may not contain the element we are looking for. As usual, include all the invariants necessary to ensure that index r does not go out of bounds, include a VARiant to prove termination, and include (as an invariant) a formula that proves correctness by asserting that upon termination, if v is in the array, then r contains its position. Hints:

   - You are likely to have **two** different finish events, one for the case in which v is in the array and another one for the case it is not.
   - I recommend having a separate Boolean variable to register whether the search has finished. This variable can then be used, e.g., in an invariant to implement a correctness condition. See Section 5.

2. Do the same for the case of a sorted array: create the model search_bin_maybe_v_m2; make it see the context search_sorted_c2, which states that f is sorted, but that v is in the array. The machine should model a a search strategy similar to search_bin_m2.

Include all the invariants necessary to ensure that index r stays within the search subwindow, include a VARiant to prove termination, and include (as an invariant) a formula that proves correctness by asserting that upon termination, if v is in the array, then r contains its position.

Hints:

- You are likely to have more than one finish event.
- I recommend using a Boolean variable to register whether the search has finished.
- In the guards of the events you will likely need to refer to v being within (or outside) the $p..q$ window.

4 Turning in the Homework

Export the final model into a .zip file (see instructions at the Rodin Tips page in the course web site and the remarks below) and **send** it to me.

Before sending me the exported model, please check that you have exported it correctly:

- Export the project.
- **Temporarily** rename the project you were working on to avoid name clashes.
- Import the project you just saved.
- Expand the machines, check that all POs have been discharged.
- Remove the project you have just loaded. In the dialog that appears when you select Remove, make sure you select the “Erase copy on disk” checkbox.

5 Booleans

Rodin / Event-B includes a built-in data type called BOOL. It does not have any specific properties or operations: it is simply a finite set (an enumerated type) that defines two constants

$$\mathsf{BOOL} = \{\mathsf{TRUE}, \mathsf{FALSE}\}$$

and the only operations that can be done are declaration, assignment, and comparison:

$b \in \mathsf{BOOL}$	b is a Boolean
$b := \mathsf{TRUE}$	Assignment
$b = \mathsf{TRUE} \Rightarrow x > 0$	Use in a logic formula
$b \Rightarrow x > 0$	Wrong: b does not have a truth value