Formal Methods in Software Development
Exercise 8 (December 22)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

December 1, 2014

The result is to be submitted by the deadline stated above *via the Moodle interface* of the course as a *.zip or .tgz* file which contains

1. a PDF file with
   - a cover page with the course title, your name, Matrikelnummer, and email address,
   - the deliverables requested in the description of the exercise,
2. the JML-annotated Java files developed in the exercise,
3. the proof files generated by the KeY prover (use the menu option “Save”).

Exercise 8: JML Specification & Verification

We consider the problem of rotating the elements of an array a by d positions to the "left"; e.g., if $a = [1, 2, 3, 4, 5, 6, 7]$ and $d = 2$, then we have after the rotation $a = [3, 4, 5, 6, 7, 1, 2]$.

The problem is supposed to be solved by the method `rotateLeft(a,d)` in the attached class `Exercise8` by three calls of method `revert(a,p,n)` that reverts the segment of a that starts at position p and has length n. First, the leading d elements are reverted, yielding for above input $a = [2, 1, 3, 4, 5, 6, 7]$. Then, the trailing $n - d$ elements are reverted, yielding $a = [2, 1, 7, 6, 5, 4, 3]$. Then the whole array is reverted yielding the result $a = [3, 4, 5, 6, 7, 1, 2]$.

Formalize the specifications of the methods `rotateLeft`, and `revert` in the attached class `Exercise8` in the JML *heavy-weight* format by a precondition (`requires`), frame condition (`assignable`), and postcondition (`ensures`). Furthermore, give the class a function `main` that allows you to test the code by some calls of method `rotateLeft`.

First use `jml` to type-check the specification. Then use the JML runtime assertion compiler `jmlc` and assertion checker `jmlrac` (respectively `openjmlrun`) to validate the specification by at least five calls of `rotateLeft` including a null array, an array of length zero, an array of length one, an array of length two, and a longer array. Then use `escjava2` to further validate your specification (which may or may not give warnings which you may or may not ignore).

When you are confident with your specifications, verify the method `rotateLeft` with KeY.

Now annotate the loop in `revert` with an appropriate invariant (`loop_invariant`) and termination term (`decreases`) and repeat the checks. When you are confident about these annotations, provide the loop also with an `assignable` clause (which is not standard JML but needed by the KeY prover). Then verify the method with KeY. This proof is independent of the proof of `rotateLeft`, so it may succeed, even if the other one failed (and vice versa).

> Recommendation: you may split a loop invariant into multiple `loop_invariant` statements; then it is easier to determine which part of an invariant failed. Do not forget to specify the bounds for `left` and `right` and the unique functional relationship between both variables. Also do not forget to specify which parts of the array have remainained unchanged so far.

If your annotations are correct and sufficiently strong, the proofs should run through automatically with less than 10,000 steps of the KeY prover (be sure that in tab "Proof Search Strategy" the "Defaults" options are selected; you may want to reduce the "Max. Rule Applications" to speed up the proof search). If you cannot complete the proof, investigate the proof tree to find out what went wrong and reconsider your annotations.

The deliverables of this exercise consist

- a nicely formatted copy of the JML-annotated Java code used for the following checks,
- the output of running `jml -Q` on the class,
- the output of running `jmlrac/openjmlrun` on the class,

- the output of running `escjava2 -NoCautions` on the class,
- a nicely formatted copy of the JML-annotated Java code used for running the KeY prover,
- for each proof, a screenshot of the KeY prover when the proof has been completed (respectively with an open state if you could not complete the proof),
- for each proof, an explicit statement where you say whether you could complete the KeY proof or not (and how many states have remained open) and optionally any explanations or comments you would like to make.