Formal Methods in Software Development
Assignment 4 (December 8)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

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,
   - a section for each part of the exercise with the requested deliverables and optionally any explanations or comments you would like to make;
2. the RISCAL specification (`.txt`) file(s) used in the exercise.

Email submissions are *not* accepted.

Assignment 4: Specifying and Verifying Procedures

Consider the problem of sorting an integer array a of length $N \geq 1$ in ascending order.

In the attached RISCAL file you find a procedure `sort(a)` that solves this problem by the *selection sort* algorithm: down-word iterating $i := N, N-1, \ldots, 2$, the procedure determines (by the call `maximum(a,i)` of an auxiliary procedure) a position j of the maximum of the first i values of a and swaps $a[j]$ with $a[i-1]$. After decrementing i, this ensures that a is sorted from position i on and that, if $i < N$, all elements at smaller positions are smaller than $a[i]$. Thus, when finally $i = 2$, a is sorted.

For each of the procedures `maximum` and `sort` perform the following tasks:

1. Specify the procedure by pre-and post-conditions. Validate the specification by executing the procedure for all legal inputs.

   For the purpose of this exercise, in specification of `sort` is suffices to require that the result array is sorted; it is not necessary to specify that this array is actually a permutation of a.

2. Validate the specification by checking (for small values of array length $N \geq 4$ and maximum element size $M \geq 3$) the tasks under header "verify specification preconditions" and "validate specification" (make sure to switch off option "Silent" when investigating the output of "Execute specification"). For checking the theorems you may use "Apply SMT solver".

3. Annotate the loop in the procedure by invariants and termination measures. Validate the annotations by running the procedure again, thus demonstrating that they are not too strong.

   For determining the invariants in the `sort` loop, take above brief argument for the correctness of the algorithm into account.

4. Check all verification conditions related to the correctness of the implementation ("verify specification preconditions", "verify implementation preconditions", "is result correct?", "verify iteration and recursion"), thus demonstrating that the annotations are strong enough.

5. Prove all verification conditions related to the correctness of the implementation with both methods SMT and MESON, thus demonstrating the correctness of the procedure for arbitrary values of N and M.

   For both proof method SMT and proof method MESON, it should be possible to prove all (sub)problems related to method `maximum`; this may, however, not be the case for method `sort` (in the sample solution, only proof method MESON was able to prove all conditions). Please report on your experience.

Please note that the entry "Apply SMT Solver" in the pop-up menu of a verification condition only checks the condition for *fixed* values of N and M; to actually prove the condition for *arbitrary* values of N and M, one has to select from the top-level menu "TP" one of the methods "SMT" or "MESON" and then select the entry "Apply Theorem Prover" in the pop-up menu of the condition.

The deliverables of this assignment consist of

1. a nicely formatted copy of the RISCAL specifications (included as text, not as screenshots);
2. for both procedures, screenshots of the RISCAL user interface showing the results of all tasks after checking (should be all blue);
3. explicit statements whether the checks of all verification conditions succeeded, whether their proofs succeeded with method SMT, and whether their proofs succeeded with method MESON.
4. for procedure `sort` and *one* subproblem of type *Is invariant preserved* that could not be solved in MESON by automatic decomposition but by proof search, a reasonably detailed explanation of the derivation of this subproblem and its proof (with both options "SMT: Med" and "SMT: Max") illustrated by screenshots (pop-up menu entry "Open Theorem Prover GUI").