Formal Methods in Software Development
Assignment 4 (December 9)

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 following problem: given an array a of length N, compute an array b of length N and a value $n \leq N$ such that:

- all the elements of b at the first n positions are different.
- every element that occurs in a occurs in b at some of the first n positions.
- every element that occurs in b at the first n positions occurs in a.

Thus b represents a "compacted" version of a in which no duplicates occur; n represents the number of distinct elements in a.

In the attached RISCAL file you find a procedure `toset(a,n)` whose `result` is a record where `result.a` is the compacted version of a and `result.n` is the number of distinct elements. This procedure calls an auxiliary procedure `has(a,n,x)` that returns true if and only if value x occurs amont the first n elements of array a.

For each procedure `has` and `toset` perform the following tasks:

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

   Please note that procedure `has` essentially implements a linear search in a; take this as the inspiration for its specification.

2. Validate the specification by checking (for small values of array length N and maximum element size M) 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.

   In the loop invariant of procedure `toset`, specify (along with all necessary range conditions on variables *ia* and *ib*), appropriate generalizations of the three output conditions given above by considering which part of the array has been already processed before the current loop iteration. As for the loop invariant of procedure `has`, take the corresponding loop invariants for linear search as an inspiration.

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.

   In some MESON proofs related to `has`, it may be necessary to switch off in the prover GUI the option "single goal" (or alternatively switch off the option "use type bounds"); in some MESON proofs related to `toset`, it may be necesssary to switch off the option "use type bounds" (and potentially also increase the "timeout" value to 60s or so).

   For both proof method SMT and proof method MESON, it should be possible to prove all (sub)problems related to method `has` However, probably neither proof method is able to prove all (sub)problems releated to procedure `toset`; please report which could not be proved (SMT probably leaves one problem open, MESON probably 2–4, all related to the preservation of some quantified invariant).

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 `toset` and *one* subproblem of type *Is invariant preserved* that could not be solved 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").