Formal Methods in Software Development
Exercise 1 (October 30)

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;
3. the RISC ProofNavigator (`.pn`) file(s) used in the exercise;
4. the proof directories generated by the RISC ProofNavigator.

Email submissions are *not* accepted.

Exercise 1a: Validating and Checking Program Specifications

(65P) We consider the following two problems:

1. Given an array a of positive length N that holds only non-negative integers, find the maximum m of a, i.e., the value m that occurs in a (i.e., at some index of a) and that is greater than or equal to all values of a (i.e., the values at all indices of a).
2. Given an array a of positive length N that holds only non-negative integers, find the index p of the maximum of a, i.e., a non-negative integer p less than n such that the element at p is greater than or equal to all values of a (i.e., the values at all indices of a).

In the RISCAL specification file `maximum.txt` you find the definitions of two procedures `maximumValue` and `maximumIndex` that solve these problems, respectively. The specification is based on two integer types `index` and `elem` that bound the domain of possible array indices respectively values by constants N and M respectively; for checking, these constants may be set independently to moderately large values.

For *each* of the procedures perform the following tasks (the definitions already given in the specification file must *not* be modified):

(a) Formalize it preconditions and postconditions as predicates (if a condition is a conjunction, it is recommended to use separate predicates for each conjunct).

In the formulation, do not use the arithmetic quantifier `max` but only the predicate logic quantifiers ∀ and ∃ (translate above specification from natural language to logic). Hint: a formula $(\forall v{:}T \texttt{ with } F.\ G)$ is equivalent to $(\forall v{:}T.\ F \Rightarrow G)$ and a formula $(\exists v{:}T \texttt{ with } F.\ G)$ is equivalent to $(\exists v{:}T.\ F \wedge G)$; the former notation may be preferred.

(b) Formulate a theorem that states that, for every input that satisfies the precondition, there exists some output that satisfies the postcondition, and check that theorem.

(c) Formulate a theorem that states that, for every input that satisfies the precondition, not every output satisfies the postcondition, and check that theorem.

(d) Formulate a theorem that states that, for every input that satisfies the precondition, the output is uniquely defined by the postcondition, and check that theorem.

(e) Use the precondition and postcondition to implicitly define a function and check whether the computed results are as desired.

(f) Annotate the procedure with preconditions and check the correctness of the procedure for all possible inputs; if a condition is a conjunction, it is recommended to use multiple annotation clauses.

Perform all checks with both $N = 0$ and some value $N > 0$.

The deliverables for this exercise consists of the following items:

1. a nicely formatted copy of the extended specification (included as text, not as screenshots);

2. the outputs of the checks (included as text, not as screenshots);
3. if the check gives an error, an explanation of the error and a justified statement that describes whether this indicates an error in your specifications or not.

Exercise 1b: Computer-Supported Predicate Logic Proofs

(35P) Take the file `exercise1b.pn` and use the RISC ProofNavigator to prove the formulas A, B, and C in this file.

The proofs only require the commands `scatter`, `split`, and `instantiate`. The proof of formula C can be considered a “proof by contradiction”: after scattering, the command `flip` may be used to introduce the negation of the goal as an assumption.

For developing the proofs, you may also try `auto`; the submitted proofs, however, must *not* make use of the `auto` command. Please also try the repeated application of the command `flatten` (rather than `scatter`) to see the gradual decomposition of the proof.

The deliverables for this exercise consists of the following items:

1. a (nicely formatted) copy of the ProofNavigator file (included as text, not as screenshots);
2. for each proof of a formula F, a readable screenshot of the RISC ProofNavigator after executing the command `proof` F,
3. an explicit statement whether the proof succeeded.