Formal Methods in Software Development
Exercise 1 (November 9)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

October 2, 2009

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

- A PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - the derivation of the verification conditions (not only the finally generated conditions),
  - a listing of the ProofNavigator file used in the exercise,
  - for each proof of a formula F, a screenshot of the RISC ProofNavigator after executing the command `proof` F,
  - optionally any explanations or comments you would like to make;
- the RISC ProofNavigator (.pn) file used for the proof;
- the proof directory `exercise1` generated by the RISC ProofNavigator.

Exercise 1: Overwriting Elements

Take the Hoare triple

$$\begin{array}{l}\{a = olda \wedge p = oldp \wedge q = oldq \wedge x = oldx \wedge \\ \quad length(a) = n \wedge length(b) = n \wedge \\ \quad 0 \leq p < n \wedge 0 \leq q \leq n \wedge i = 0\}\end{array}$$

```
while i < n do
  if p <= i and i < q then
    b[i] := x
  else
    b[i] := a[i];
  end
  i := i+1;
end
```

$$\begin{array}{l}\{a = olda \wedge p = oldp \wedge q = oldq \wedge x = oldx \wedge \\ \quad length(a) = n \wedge length(b) = n \wedge \\ \quad (\forall j : 0 \leq j < p \Rightarrow b[j] = a[j]) \wedge \\ \quad (\forall j : p \leq j < q \Rightarrow b[j] = x) \wedge \\ \quad (\forall j : q \leq j < n \Rightarrow b[j] = a[j])\}\end{array}$$

which describes the core proof obligation for a program that fills into array b a copy of array a with value x overwriting the values at all positions in the interval $[p \ldots q[$.

The (partial) correctness of this Hoare triple can be verified with the help of the following loop invariant:

$$\begin{array}{l}a = olda \wedge p = oldp \wedge q = oldq \wedge x = oldx \wedge \\ length(a) = n \wedge length(b) = n \wedge \\ 0 \leq p < n \wedge 0 \leq q \leq n \wedge 0 \leq i \leq n \wedge \\ \forall j : 0 \leq j < i \;\Rightarrow \\ \quad (j < p \Rightarrow b[j] = a[j]) \wedge \\ \quad (p \leq j \wedge j < q \Rightarrow b[j] = x) \wedge \\ \quad (q \leq j \Rightarrow b[j] = a[j])\end{array}$$

Your first task is to use this invariant to produce the four verification conditions for proving the partial correctness of the program (one for showing that the input condition implies the invariant, one for showing that the invariant and the negation of the loop condition implies the output condition, two for showing that the invariant is preserved for each possible execution path in the loop body).

Your second task is to verify these four conditions in the style of the verification of the "linear search" algorithm presented in class with the help of the RISC ProofNavigator. For this, write a declaration file with the following structure:

```
newcontext "exercise1";

// arrays as presented in class
...

// program variables and mathematical constants
a: ARR; olda: ARR;
p: NAT; oldp: NAT;
...

// predicates
...

// verification conditions
...
```

First define predicates `Input`, `Output`, and `Invariant`, where (as shown in class) `Invariant` is parameterized over the program variables. Then define four formulas `A`, `B1`, `B2`, `C` for the verification conditions and prove these.

All proofs can be done with the command `expand`, `scatter`, `instantiate`, `split`, and `auto` only (actually, not even all of these are required). Various proof branches can be closed by (simultaneous) expansion of the definitions of `get`, `put`, `content`, `length`.