Formal Methods in Software Development
Exercise 3 (November 24)

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

October 27, 2008

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 specification of the code and 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 generated by the RISC ProofNavigator.

Exercise 3: A Verification

Take the following piece of code code

```
i = l; j = r;
while (i <= j)
{
  if (a[i] < v)
    i = i+1;
  else if (a[j] >= v)
    j = j-1;
  else
  {
    t = a[i]; a[i] = a[j]; a[j] = t;
    i = i+1; j = j-1;
  }
}
m = i;
```

which describes the *partitioning phase* of the Quicksort algorithm: given an array a, indices $l \leq r$ within that array, and a partitioning value v, the program reshuffles the array $a[l \ldots r]$ such that, for some index m with $l \leq m \leq r$, $a[l \ldots m-1]$ holds all elements less than v and $a[m \ldots r]$ holds all the others.

First develop an adequate specification of this piece of code as a Hoare triple. Please note that, among other conditions, this specification must also state that after the execution of the program the array holds the same elements as before (but possibly in a different order). You may use for this purpose the (undefined) predicate $same(a, b, n)$ which shall express the fact that arrays a and b have the same elements in the first n positions (but possibly in a different order).

Then use strongest postcondition/weakest precondition reasoning to boil down the verification to the verification of a Hoare triple for the while loop.

Now find a suitable loop invariant and a suitable termination term, and produce from these the conditions for verifying the *total* correctness of the triple. Please note in the definition of the loop invariant that the index j may become less than l and might ultimately equal -1; likewise the index i may become greater than r and might ultimately equal $length(a)$.

Finally, prove these conditions with the help of the RISC ProofNavigator (where you may define $same(a, b, n) :\Leftrightarrow$ true). The proof proceeds by scattering, expansion of definitions, manual and/or automatic instantiations. Only in the verification condition corresponding to the third conditional branch in the loop body, the definitions of the array operations have to be expanded for closing some branches in the proof.