Interactive Software Verification SS 2013

https://www21.in.tum.de/teaching/isv/SS13

H. Gast, L. Noschinski

SHEET 6

Proofs with Abstractions

Date: 11.6.2013
Hand-In: 18.06.2013, 08:30

Goals You get a better overview over correctness proofs and actively use abstractions.

Before Use the same setup of SimplC as in the previous exercise.

Exercise 1 [4] Copy of an Array

The following code obviously copies all elements of an array `a` into an array `b`.

```
i = 0;
while (i < n) {
    b[i] = a[i];
    i = i + 1;
}
```

Prove that this is true.

1. Write down the appropriate pre- and postconditions. Use the elems abstractions from the lecture.
2. Specify the loop invariant and verify the code.

Exercise 2 [6] Reverse an Array

The following code reverses the elements of an array (analogue to `partition` from the lecture).

```
i =0;
j = n;
while (i < j) {
    t = a[i];
    a[i] = a[j-1];
    a[j-1] = t;
    i = i + 1;
    if (i < j)
        j = j - 1;
}
```

Prove that the precondition

n $\leq$ alength a $\wedge$ elems a 0 n = A $\wedge$ 0 $\leq$ n

implies the following postcondition:

elems a 0 n = rev A

For the main part of the invariant, you can use

A = rev (elems a j n) @ elems a i j @ rev (elems a 0 i)

Hint: For this exercise, `simp add: elems_split_first` does to much. Use `subst` to apply the lemmas `elems_split_first` and `elems_split_last` in a controlled way. Solve the conditions with `assumption` and `simp`.