Interactive Software Verification SS 2013

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

H. Gast, L. Noschinski

SHEET 10

Date: 15.01.2013

Hand-In: 22.1.2013, 23:50

Lists in Separation Logic

Goals You can use Separation Logic for proofs over lists.

Exercise 1 [5] Length of Lists

The following program obviously computes the length of a singly-linked list on the heap.

```
n = 0;
t = p;
while (t != null) {
    n = n + 1;
    t = t->next;
}
```

This is expressed by the following specification:

pre: "(list p XS NULL) heap"
post: "(list p XS NULL · n = of-nat (length XS)) heap"

Invent an appropriate loop invariant and prove the correctness.

Exercise 2 [5] Symbolic execution

The methods `step` and `run` perform symbolic execution. To see what steps need to be performed by these methods, prove the correctness of the following simple program, without using the `step`, `run` and `heap` methods:

```
t = *p;
*p = *q;
*q = t;
```

The specification is:

pre: "(p → cty-int. X ⋆ q → cty-int. Y) heap"
post: "(p → cty-int. Y ⋆ q → cty-int. X) heap"

Hints:

- Most of this proof steps can be performed with introduction rules. You can search for an introduction rule matching your current goal with `find_theorems intro`.
- Lemmas to rearrange spatial assumptions are found in `test_direct_matching`.
- To make your proof nicer (after you have finished it): Isabelle supports backtracking for proof methods. For example, if `rule` can perform different steps (e.g., because it was given a list of theorems), you can use the `back` command to make it backtrack and use the next option. If you chain multiple proof methods together by `apply (method_1, method_2, ..., method_n)`, Isabelle backtracks automatically if one of the proof methods fails.