Exercise Sheet 2: Specification and Verification with Higher-Order Logic (Summer Term 2010)

Date: 19.04.2010

Exercise 1 Functions in ML

Please do not use the append operator `@` or any other predefined functions on lists for this exercise. (You can use `foldl` for h) and i), if you like.)

a) (Prepare!) Write a function `swap : 'a * 'b -> 'b * 'a`, which swaps the two components of a pair.

b) (Prepare!) Write a function `listSwap : ('a * 'b) list -> ('b * 'a) list`, which swaps all pairs of a list.

c) Write a function `map : ('a -> 'b) -> 'a list -> 'b list`, which applies a function to all elements of a list.

d) Write a function `listSwap2 : ('a * 'b) list -> ('b * 'a) list`, with the same behavior as `listSwap`, using the `map` function instead of recursion.

e) (Prepare!) Write a function `appendRight : 'a list -> 'a -> 'a list`, which appends a single element at the end of a list.

f) (Prepare!) Write a function `reverse : 'a list -> 'a list`, which reverts a list.

g) Write a function `replace : 'a -> 'a -> 'a list -> 'a list`, where the call `replace x y l` should return a list in which all occurrences of `x` in `l` are replaced with `y`.

h) Write a function `forall : ('a -> bool) -> 'a list -> bool`, which calculates wether all elements of a list satisfy the given predicate.

i) Write a function `exists : ('a -> bool) -> 'a list -> bool`, which calculates wether any element of a list satisfies the given predicate.

For additional exercises in ML, please refer to the slides, exercise sheets and solutions of the lecture “Software-Entwicklung I” from the winter term 2008/09. (In the winter 2009/10 we did Haskell in the beginners course.)

You can also find a lot of suggestions for functions on lists in the documentation of the Haskell Data.List module [http://haskell.org/ghc/docs/latest/html/libraries/base-4.2.0.0/Data-List.html](http://haskell.org/ghc/docs/latest/html/libraries/base-4.2.0.0/Data-List.html).

Please do not hesitate to ask us, if you encounter any problems when implementing such functions.
Exercise 2 Functions and Properties in Isabelle/HOL

a) Define the functions appendRight, reverse and replace as primitive recursive functions in Isabelle/HOL. Now prove or disprove the following properties:

- \( \text{replace } x \ y \ (\text{replace } x \ y \ l) = \text{replace } x \ y \ l \)
- \( y \neq x \rightarrow \text{replace } x \ z \ (\text{replace } x \ y \ l) = \text{replace } x \ y \ l \)
- \( \text{replace } y \ z \ (\text{replace } x \ y \ l) = \text{replace } x \ z \ l \)
- \( \text{reverse } (\text{replace } x \ y \ l) = \text{replace } x \ y \ (\text{reverse } l) \)

b) Define the functions forall and exists as primitive recursive functions in Isabelle/HOL. Then prove the following properties:

- \( \forall x. \ P \ x \land Q \ x \ l = (\forall P \ l \land \forall Q \ l) \)
- \( \exists x. \ P \ x \lor Q \ x \ l = (\exists P \ l \lor \exists Q \ l) \)
- \( \exists P \ (\text{map } f \ l) = \exists (P \circ f) \ l \)
- \( \forall P \ (\text{reverse } l) = \forall P \ l \)
- \( \exists P \ (\text{reverse } l) = \exists P \ l \)

Exercise 3 Insertion Sort in Isabelle/HOL

a) Specify the sorting algorithm “insertion sort” on lists of integers as primitive recursive function

\[ \text{insertionSort :: 'a list => 'a list} \]

in Isabelle/HOL. The elements of the resulting list should be in ascending order.

b) Write a predicate

\[ \text{sortedPerm :: 'a list => 'a list => bool} \]

which takes two lists and tests whether the first is a sorted permutation of the second.

c) Proof the implementation of \text{insertionSort} correct, regarding the predicate \text{sortedPerm}.