Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. J. O. Blech
Dipl.-Inform. M. J. Gawkowski

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Handout 5: Specification and Verification Using Higher-Order Logic (summer term 2008)

May 20—th, 2008

Objectives and lerning targets of this exercise:
In this exercise we want to

1. complete formal framework on specification and verification of the quicksort algorithm presented during the lecture and
2. investigate alternative correctness specifications of the quicksort algorithm.

After accomplishing this exercise you should be able to conduct nontrivial proofs by induction on types that are most commonly used in HOL verification, sets and lists. Further, you lern how to set up small correctness specifications of algorithms involving sets and lists.

Exercise 1 Quicksort

Download theory file "Quicksort.thy" from our lecture homepage and make yourself familiar with constant definitions, function definitions, and proofs provided by the theory Quicksort.

a) Complete proofs in "Quicksort" which are finished using keyword `sorry`.

b) Set up an alternative specification of sorted lists by defining a (possibly primitive) recursive function `sorted2` :: *dataset list* → *bool* and prove the following lemma:

```
lemma qsort_sort_prop2 :
"sorted2(qsorted xl)"
```

c) Set up an alternative predicate `is_perm_of` :: *datatset list* × *datatset list* specifying when two lists of `datatset` elements are each other's permutations and prove the following lemma:

```
lemma qsort_permutation_prop :
"is_perm_of(qsort l, l)"
```

Here, we provide a proposal for simplified vesion of `is_perm_of` for integer lists.

```
constdefs is_perm_of :: int list × int list
"is_perm_of ≡ λ (l₁, l₂).
      length(l₁) = length(l₂)∧
      ∀x.x mem l₁          ⟶
        length(filter (λ y.y ≤ x) l1) = length(filter (λ y.y ≤ x) l2)
        length(filter (λ y.y > x) l1) = length(filter (λ y.y > x) l2)
```