

# COMP 4161 S2/08

## Advanced Topics in Software Verification

### Assignment 3

This assignment starts on Wed, 24.09.2008 and is due on Wed, 08.10.2009, 13:30h. Please submit by email to `gerwin.klein@nicta.com.au`. Accepted formats are plain text, Isabelle theory files and pdf documents.

#### 1 Quicksort (15 marks)

Show `sorted (quicksort xs)` for the `quicksort` definition of the lecture where `sorted` is the pre-defined predicate over lists of natural numbers.

#### 2 Sets as Interval Lists

One common data structure implementation problem is that of implementing sets of numbers. In the following, we will implement sets of natural numbers (*nat set*) as an ordered list of intervals (*nat \* nat list*).

The set

$$\{2, 3, 5, 7, 8, 9\}$$

for instance should be implemented as

$$[(2, 4), (5, 6), (7, 10)]$$

That means, the intervals are inclusive on the left and exclusive on the right component. We want the implementation to have unique representatives for each set. This implies that intervals should be merged whenever possible, i.e.  $[(2, 4), (4, 5)]$  is not a valid implementation and should be written as  $[(2, 5)]$ . Intervals should be in ascending order from left to right in the list.

##### (a) (18 marks)

Define a type abbreviation `intervals` for interval lists and two constants `is_int_list` and `set_of`.

The predicate `is_int_list` should decide if a given interval list satisfies the implementation constraints above.

The function `set_of` should take an interval list and return the set of natural numbers this list represents. Use the notation  $\{a..<b\}$  for the set of numbers from  $a$  to  $b$ , excluding  $b$ .

##### (b) (20 marks)

Define two functions `add` and `rem` that add/remove one interval to/from a given list of intervals. The functions should merge intervals accordingly such that they preserve the implementation invariant `is_int_list`. Write at least three concrete test cases for each function as lemmas and execute them with `simp` or `auto`.

(c) **(22 marks)**

Prove

$$\text{set\_of } (\text{add } (a,b) \text{ } xs) = \text{set\_of } xs \cup \{a..<b\}$$

and:

`add` preserves the `is_int_list` invariant.

You may make an additional assumption on  $a$  and  $b$  and for the `set_of` lemma you may assume `is_int_list xs` if necessary.

(d) **(25 marks)**

Prove

$$\text{set\_of } (\text{rem } (a,b) \text{ } xs) = \text{set\_of } xs - \{a..<b\}$$

and:

`rem` preserves the `is_int_list` invariant.

You may make an additional assumption on  $a$  and  $b$  and for the `set_of` lemma you may assume `is_int_list xs` if necessary.