Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Simon Wimmer

WS 2016/17
25. 10. 2016

Semantics of Programming Languages

Exercise Sheet 1

Before beginning to solve the exercises, open a new theory file named `Ex01.thy` and add the the following three lines at the beginning of this file.

theory *Ex01*
imports *Main*
begin

Exercise 1.1 Calculating with natural numbers

Use the **value** command to turn Isabelle into a fancy calculator and evaluate the following natural number expressions:

"2 + (2::nat)" *"(2::nat) ∗ (5 + 3)"* *"(3::nat) ∗ 4 − 2 ∗ (7 + 1)"*

Can you explain the last result?

Exercise 1.2 Natural number laws

Formulate and prove the well-known laws of commutativity and associativity for addition of natural numbers.

Exercise 1.3 Counting elements of a list

Define a function which counts the number of occurrences of a particular element in a list.

fun *count* :: *"'a list ⇒ 'a ⇒ nat"*

Test your definition of *count* on some examples and prove that the results are indeed correct.

Prove the following inequality (and additional lemmas if necessary) about the relation between *count* and *length*, the function returning the length of a list.

theorem *"count xs x ≤ length xs"*

Exercise 1.4 Adding elements to the end of a list

Recall the definition of lists from the lecture. Define a function *snoc* that appends an element at the right end of a list. Do not use the existing append operator @ for lists.

fun *snoc* :: *"'a list ⇒ 'a ⇒ 'a list"*

Convince yourself on some test cases that your definition of *snoc* behaves as expected, for example run:

value *"snoc [] c"*

Also prove that your test cases are indeed correct, for instance show:

lemma *"snoc [] c = [c]"*

Next define a function *reverse* that reverses the order of elements in a list. (Do not use the existing function *rev* from the library.) Hint: Define the reverse of *x # xs* using the *snoc* function.

fun *reverse* :: *"'a list ⇒ 'a list"*

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

value *"reverse [a, b, c]"*
lemma *"reverse [a, b, c] = [c, b, a]"*

Prove the following theorem. Hint: You need to find an additional lemma relating *reverse* and *snoc* to prove it.

theorem *"reverse (reverse xs) = xs"*

Homework 1.1 More Finger Exercise with Lists

Submission until Tuesday, November 1, 10:00am.

Mail a theory file named FirstnameLastname01.thy (replace with your name!) which runs in Isabelle-2016 **without errors** to *wimmersatindottumdotde*.
General hints:

- If you cannot prove a lemma, that you need for a subsequent proof, assume this lemma by using sorry.
- Define the functions as simply as possible. In particular, do not try to make them tail recursive by introducing extra accumulator parameters — this will complicate the proofs!
- All proofs should be straightforward, and take only a few lines.

Define a function *spread* that spreads an element among a list. This is, *spread a xs* adds the element *a* behind every element of *xs*. The following evaluates to true, for instance:

value *"spread 0 [1,2,3] = [1,0,2,0,3,0]"*

Prove that spreading an element among a list *xs* adds exactly *length xs* copies of the element to the list.

lemma *"count (spread a xs) a = count xs a + length xs"*

Finally, prove the following lemma connecting *reverse* and *spread*:

lemma *"snoc (reverse (spread a xs)) a = a # spread a (reverse xs)"*

Hint: You may need an auxiliary lemma about *spread* and *snoc*.