Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Brian Huffman, Peter Lammich

WS 2011/12
21. 10. 2010

Semantics of Programming Languages

Exercise Sheet 1

Before beginning to solve the exercises, open a new theory file named `Ex01.thy` and write the the following three lines at the top 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 Sum of odd numbers

Submission until Wednesday, November 2, 12:00 (noon).

In this homework assignment you will prove that the square of a natural number n can be computed as the sum of the first n odd numbers, which we will write as $oddsum(n)$. For example, we have $oddsum(3) = 1 + 3 + 5 = 9 = 3 * 3$.

Your first task is to use the **fun** command to define a function *oddsum* :: *nat ⇒ nat* in Isabelle. Your definition should have equations for $oddsum(0)$ and $oddsum(Suc\ n)$.

fun *oddsum* :: *"nat ⇒ nat"*

You may wish to use the **value** command to check that your definition is correct. For example, the following command should evaluate to *True*.

value *"oddsum 3 = 1 + 3 + 5"*

Your next task is to prove by induction that for any n, $oddsum(n)$ computes the square of n. First, write an informal proof by hand. Your proof should contain a base case for

zero, where you show that *oddsum*(*0*) equals the square of *0*. Next you should have a case for successor: Fix an arbitrary m, assume the inductive hypothesis that *oddsum*(m) equals the square of m, and then show that *oddsum*(*Suc* m) equals the square of *Suc* m.

Finally, prove the same property *formally* in Isabelle:

lemma *"oddsum* $n = n * n$*"*