Semantics of Programming Languages
Exercise Sheet 1

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:

\[2 + (2::nat) \quad (2::nat) \times (5 + 3) \quad (3::nat) \times 4 - 2 \times (7 + 1)\]

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 all additionally necessary lemmas) 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 \textit{snoc} behaves as expected, for example run:

\textbf{value} “\textit{snoc} [] c”

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

\textbf{lemma} “\textit{snoc} [] c = [c]”

Prove the following theorem. Hint: you need to find an additional lemma to prove it.

\textbf{theorem} “\textit{rev} (x \# xs) = \textit{snoc} (\textit{rev} xs) x”

\textbf{Exercise 1.5} Tree traversal

Extend the tree datatype of the lecture (in \texttt{Tree Demo.thy}) in such a way that values are also stored in the leaves of a tree. Also reformulate the \textit{mirror} function accordingly.

Define functions \textit{pre-order} and \textit{post-order}, which traverse a tree and collect all stored elements in a list in the respective order, such that the following theorem holds. You may use any of the previously defined functions and may need to prove additional lemmas.

\textbf{theorem} “\textit{pre-order} (\textit{mirror} t) = \textit{rev} (\textit{post-order} t)”

\textbf{Homework 1} Leaves of a tree

\textit{Submission until Wednesday, November 3, 2010, 12:00 (noon).}

Define a datatype \texttt{ntree} of binary trees which store natural numbers in leaves, but no data in inner nodes. Moreover, write a function which returns, for such a binary tree, a list containing all natural numbers stored in the leaves, in any order and without removing duplicates.

\textbf{fun} \texttt{leaves :: “ntree ⇒ nat list”}

Additionally, define a function which counts the number of leaves in a tree.

\textbf{fun} \texttt{leaf\_count :: “ntree ⇒ nat”}

Then prove the following property about binary trees; you may need to prove additional lemmas.

\textbf{theorem} “\texttt{length (leaves t)} = \texttt{leaf\_count t}”

Now write a function \texttt{treesum} which sums up the natural numbers stored in a binary tree.

\textbf{fun} \texttt{treesum :: “ntree ⇒ nat”}

Prove the following correspondence between this function and the function \texttt{listsum}, which sums up the elements of a list.

\textbf{lemma} “\texttt{listsum (leaves t)} = \texttt{treesum t}”

2