Interactive Software Verification SS 2013

https://www21.in.tum.de/teaching/isv/SS13

H. Gast, L. Noschinski

SHEET 1

Introduction to Isabelle

Date: 17.04.2013
Hand-In: 24.04.2013, 8:30

Goals You learn to formalize simple situations and recursive functions in Isabelle and prove simple theorems about these.

Hint On this sheet, all proofs can be found by `auto`, if you use induction where necessary and give the right equations to the simplifier.

Exercise 1 [4] Functional Programming

Write a function

sumup :: "int list ⇒ int"

which computes the sum of numbers of a given Isabelle/HOL list.

Evaluate the function on concrete values to get a feeling for its correctness:

value "sumup []"
value "sumup [1]"
value "sumup [1,2]"
value "sumup [1,2,45,62]"

Prove now that the sumup function is distributive, i.e. that sumup (*xs* @ *ys*) = sumup *xs* + sumup *ys* holds.

Afterwards, prove that sumup returns the same result for reversed lists, i.e. that sumup (rev *xs*) = sumup *xs* holds

Exercise 2 [2] Nodes of binary trees

In the exercise theory, we defined a type of binary trees with functions nleaves and ninner counting the leaves and inner nodes.

Define a predicate

fun bt-full :: "bt ⇒ bool"

recognizing full binary trees. A binary tree is called *full*, iff for all Nodes either both or none of the children are Tip.

Formulate and prove the the property that for each full tree holds leaves t = ninner t + 1.

Hint: To prove this property, you will need the following lemma:

$t \neq \mathsf{Tip} \leftrightarrow (\exists\, l\ v\ r.\ \mathsf{t} = \mathsf{Node}\ \ l\ \ v\ \ r)$

This can be proved by case analysis on *t*, which can be performed with `apply (cases t)`.

Exercise 3 [4] Summing Binary Trees

Write a function

sumtree :: "bt ⇒ int"

which computes the sum of all elements of a tree and a function

list-of-tree :: "bt ⇒ int list"

which computes the list of all elements (including duplicates) of a tree.

Then prove sumtree *xs* = sumup (list-of-tree *xs*).