Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

SS 23
23. 5. 2023

Functional Data Structures

Exercise Sheet 5

Solve this exercise sheet without *sledgehammer* proofs i.e., *smt*, *metis*, *meson*, or *moura* are forbidden!

Exercise 5.1 Bounding power-of-two by factorial

Prove that, for all natural numbers $n > 3$, we have $2^n < n!$. We have already prepared the proof skeleton for you.

lemma *exp_fact_estimate*: *"n>3 ⟹ (2::nat)^n < fact n"*
proof (*induction n*)
  case *0* **then show** *?case* **by** *auto*
next
  case (*Suc n*)
  show *?case*

Fill in a proof here. Hint: Start with a case distinction whether *n>3* or *n=3*.

Warning! Make sure that your numerals have the right type, otherwise proofs will not work! To check the type of a numeral, hover the mouse over it with pressed CTRL (Mac: CMD) key. Example:

lemma *"2^n ≤ 2^Suc n"*
  apply *auto* **oops**

Leaves the subgoal $2 \mathbin{\hat{}} n \leq 2 * 2 \mathbin{\hat{}} n$

You will find out that the numeral *2* has type $'a$, for which you do not have any ordering laws. So you have to manually restrict the numeral's type to, e.g., *nat*.

lemma *"(2::nat)^n ≤ 2^Suc n"* **by** *simp*

Exercise 5.2 Sum Squared is Sum of Cubes

- Define a recursive function *sumto* $f\ n = \sum_{i=0...n} f(i)$.
- Show that $(\sum_{i=0...n} i)^2 = \sum_{i=0...n} i^3$.

fun *sumto* :: "$(nat \Rightarrow nat) \Rightarrow nat \Rightarrow nat$"

You may need the following lemma:

lemma *sum_of_naturals*: "$2 * sumto\ (\lambda x.\ x)\ n = n * (n + 1)$"

lemma "$sumto\ (\lambda x.\ x)\ n\ \hat{}\ 2 = sumto\ (\lambda x.\ x\hat{}3)\ n$"
proof (*induct n*)
  case *0* **show** *?case* **by** *simp*
next
  case (*Suc n*)
  assume *IH*: "$(sumto\ (\lambda x.\ x)\ n)^2 = sumto\ (\lambda x.\ x\ \hat{}\ 3)\ n$"
  note [*simp*] = *algebra_simps* — Extend the simpset only in this block
  show "$(sumto\ (\lambda x.\ x)\ (Suc\ n))^2 = sumto\ (\lambda x.\ x\ \hat{}\ 3)\ (Suc\ n)$"

Exercise 5.3 Pretty Printing of Binary Trees

Binary trees can be uniquely pretty-printed by emitting a symbol L for a leaf, and a symbol N for a node. Each N is followed by the pretty-prints of the left and right tree. No additional brackets are required!

datatype *'a tchar* = *L* | *N 'a*

fun *pretty* :: "*'a tree* ⇒ *'a tchar list*"
value "*pretty* (*Node* (*Node Leaf 0 Leaf*) (*1::nat*) (*Node Leaf 2 Leaf*)) = [*N 1*, *N 0*, *L*, *L*, *N 2*, *L*, *L*]"

Show that pretty-printing is actually unique, i.e. no two different trees are pretty-printed the same way. Hint: Auxiliary lemma.

lemma *pretty_unique*: "$pretty\ t = pretty\ t' \Longrightarrow t=t'$"

Define a function that checks whether two binary trees have the same structure. The values at the nodes may differ.

fun *bin_tree2* :: "*'a tree* ⇒ *'b tree* ⇒ *bool*"

While this function itself is not very useful, the induction principle generated by the function package is! It allows simultaneous induction over two trees:

print_statement *bin_tree2.induct*

Try to prove the above lemma with that new induction principle.

Homework 5.1 Identity for a Summation

Submission until Monday, 29 May, 23:59pm.

The identity $\sum_{i=1\ldots n} i2^i = n2^{n+1} - (2^{n+1} - 2)$ holds for summations. Prove that identity for the function *sumto* from the tutorial

lemma *sum_ident*:
  "sumto (λi. i∗ 2 ^ i) n = n ∗ 2 ^ (n + 1) − (2 ^ (n + 1) − 2)"

Your proof should be by induction on n. In the induction step, show the derivation of the equality by a chain of equations. For readability, each step in that chain should be very simple: it must use simp or auto with at most one deleted fact and one additional fact (i.e. global lemma, assumption, IH) or *algebra_simps*. Hints:

- Try to come up with the chain of equations on paper before formalising it in Isabelle/HOL.
- The lemma *add_diff_assoc2* might be useful for the proof. One way to use that lemma is by proving its side condition before you start the chain of equalities and then using that and then adding *add_diff_assoc2*[*OF side*] to the simp set.
- Sometimes it is useful to get controlled behaviour of *simp* by having one lemma, say *lem*, as the only rule to be used by *simp*. This can be done by *simp only*: *lem*.

Homework 5.2 Estimate for Number of Leafs

Submission until Monday, 29 May, 23:59pm. Note: Use Isar, proofs using *metis*, *smt*, *meson*, *or moura* (as generated by sledgehammer) are forbidden!

Define a function to count the number of leafs in a binary tree:

fun *num_leafs* :: *"'a tree ⇒ nat"*

Start by showing the following auxiliary lemma:

lemma *auxl*:
  assumes *IHl*: *"num_leafs l ≤ 2 ^ height l"*
    and *IHr*: *"num_leafs r ≤ 2 ^ height r"*
    and *lr*: *"height l ≤ height r"*
  shows *"num_leafs(Node l x r) ≤ 2 ^ height(Node l x r)"*

Also show the symmetric statement. Hint: Copy-paste-adjust!

lemma *auxr*:
  assumes *IHl*: *"num_leafs l ≤ 2 ^ height l"*
    and *IHr*: *"num_leafs r ≤ 2 ^ height r"*
    and *rl*: *"¬ height l ≤ height r"*
  shows *"num_leafs(Node l x r) ≤ 2 ^ height(Node l x r)"*

Finally, show that we can estimate the number of leafs in a tree as follows:

theorem *num_leafs_est*: "*num_leafs t* $\leq$ *2^height t*"
proof (*induction t*)
  case *Leaf* **show** *?case* **by** *auto*
next
  case (*Node l x r*)
  assume *IHl*: "*num_leafs l* $\leq$ *2 ^ height l*"
  assume *IHr*: "*num_leafs r* $\leq$ *2 ^ height r*"
  show "*num_leafs* $\langle l,\ x,\ r\rangle \leq$ *2 ^ height* $\langle l,\ x,\ r\rangle$"

Fill in your proof here