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

SS 2022
27. 5. 2022

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

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 Landau Notation

Submission until Thursday, June 2, 23:59pm.

(**Solve the homework without** *sledgehammer* **proofs!**)

We define a (slightly simplified) version of the landau symbol $\mathcal{O}$:

$\mathcal{O}\ g = \{f.\ \exists\, c{>}0.\ \exists\, x_0.\ \forall\, x{\geq}x_0.\ f\ x\ \leq\ c * g\ x\}$

Show that $2n \in \mathcal{O}(n^2)$. Use Isar proof patterns, and make sure that your types are correct.

lemma *lin_in_square*: "$(\lambda n.\ 2{*}n) \in \mathcal{O}\ (\lambda n.\ n\hat{\ }2)$"
  unfolding *$\mathcal{O}$_def*

Show that the other direction does not hold, i.e., $n^2 \notin 2n$

Hint: to simplify quadratic formulae, give *power2_eq_square* and *algebra_simps* to the simplifier.

lemma *square_notin_lin*: "$(\lambda n.\ n\hat{\ }2) \notin \mathcal{O}\ (\lambda n.\ 2{*}n)$"

Homework 5.2 Interleave Lists

Submission until Thursday, June 2, 23:59pm.

(**Solve the homework without** *sledgehammer* **proofs!**)

The function *splice* takes two lists and interleaves them. Check its recursion equations:

thm *splice.simps*

Show that, using the splice function, every list can be constructed from two lists, where each of which is at least as long as half the length of the constructed list.

lemma *split_splice*:
  "$\exists\, ys\ zs.\ xs = splice\ ys\ zs \wedge length\ ys \geq (length\ xs)\ div\ 2 \wedge length\ zs \geq (length\ xs)\ div\ 2$"

Hint: To prove that theorem, you will need a stronger induction hypothesis than that which you get by using structural induction on lists. To get such a stronger hypothesis, you will need to use a different induction principle, like the one below.

$[\![P\ [];\ \bigwedge x.\ P\ [x];\ \bigwedge x\ y\ zs.\ P\ zs \Longrightarrow P\ (x\ \#\ y\ \#\ zs)]\!] \Longrightarrow P\ xs$

In particular, your proof should begin by *proof*(*induction xs rule*: *induct_pcpl*).