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

SS 2021
14. 5. 2021

Functional Data Structures

Exercise Sheet 5

Solve this exercise sheet without using *sledgehammer*! Proofs using *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 $2 \ \hat{}\ n \leq 2 * 2 \ \hat{}\ n$

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 Split Lists

Submission until Thursday, May 20, 23:59pm. Recall: Proofs using *metis*, *smt*, *meson*, *or moura* (as generated by sledgehammer) are forbidden! Write your proofs down structured. You should not use *apply*.

Show that every list can be split into a prefix and a suffix, such that the length of the prefix is *1/n* of the original lists's length. (This works without the assumption that *0 < n*, since division by zero is defined as zero.)

theorem *split_list*: *"∃ys zs. length ys = length xs div n ∧ xs=ys@zs"*

Homework 5.2 Estimate Recursion Equation

Submission until Thursday, May 20, 23:59pm.

Recall: Proofs using *metis*, *smt*, *meson*, *or moura* (as generated by sledgehammer) are forbidden! Write your proofs down structured. You should not use *apply*.

Show that the function defined by $a\ 0 = 0$ and $a\ (n+1) = (a\ n)^2 + 1$ is bounded by the double-exponential function *2^(2^n)*

We have given you a proof skeleton, setting up the induction. To complete your proof, you should come up with a chain of inequations.

Hint: It is a bit tricky to get the approximation right. We strongly recommend to sketch the inequations on paper first.

Hint: Have a look at the lemma *power_mono*, in particular its instance for squares:

thm *power_mono*[**where** *n=2*]

theorem *a_bound*: *"a n ≤ 2 ^ (2 ^ n) − 1"*
proof(*induction n*)
  case *0* **thus** *?case* **by** *simp*
next
  case (*Suc n*)
  assume *IH*: *"a n ≤ 2 ^ 2 ^ n − 1"*
  show *"a (Suc n) ≤ 2 ^ 2 ^ Suc n − 1"*