Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich

SS 2018
11. 5. 2017

Functional Data Structures

Exercise Sheet 5

- Import *Complex_Main* and *~~/src/HOL/Library/Tree*
- For this exercise sheet (and Homework 1), you are not allowed to use sledgehammer! Proofs using the *smt*, *metis*, *meson*, *or moura* methods 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*)
  assume *IH*: *"3 < n ⟹ (2::nat) ^ n < fact n"*
  assume *PREM*: *"3 < Suc n"*
  show *"(2::nat) ^ Suc n < fact (Suc n)"*

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

qed

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 ^ n ≤ 2 * 2 ^ 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* — Note: Type inference will infer *nat* for the unannotated numeral, too. Use CTRL+hover to double check!

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*: "$2 * sumto\ (\lambda x.\ x)\ n = n * Suc\ n$"
  by (*induction n*) *auto*

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 "$(sumto\ (\lambda x.\ x)\ (Suc\ n))^2 = sumto\ (\lambda x.\ x\ \hat{}\ 3)\ (Suc\ n)$"

Insert a proof here

qed

Exercise 5.3 Paths in Graphs

A graph is described by its adjacency matrix, i.e., $G :: 'a \Rightarrow 'a \Rightarrow bool$.

Define a predicate *path G u p v* that is true if p is a path from u to v, i.e., p is a list of nodes, not including u, such that the nodes on the path are connected with edges. In other words, $path\ G\ u\ (p_1 \ldots p_n)\ v$, iff $G\ u\ p_1$, $G\ p_i\ p_{i+1}$, and $p_n = v$. For the empty path (n=*0*), we have u=v.

fun *path* :: "$('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'a\ list \Rightarrow 'a \Rightarrow bool$"

Test cases

definition "$nat\_graph\ x\ y \longleftrightarrow y{=}Suc\ x$"
value ‹*path nat_graph 2* [] *2*›
value ‹*path nat_graph 2* [*3*,*4*,*5*] *5*›
value ‹$\neg$ *path nat_graph 3* [*3*,*4*,*5*] *6*›
value ‹$\neg$ *path nat_graph 2* [*3*,*4*,*5*] *6*›

Show the following lemma, that decomposes paths. Register it as simp-lemma.

lemma *path_append*[*simp*]: "$path\ G\ u\ (p1@p2)\ v \longleftrightarrow (\exists\, w.\ path\ G\ u\ p1\ w \wedge path\ G\ w\ p2\ v)$"

Show that, for a non-distinct path from u to v, we find a longer non-distinct path from u to v. Note: This can be seen as a simple pumping-lemma, allowing to pump the length of the path.

Hint: Theorem *not_distinct_decomp*.

lemma *pump_nondistinct_path*:
  assumes *P*: *"path G u p v"*
  assumes *ND*: *"¬distinct p"*
  shows *"∃ p'. length p' > length p ∧ ¬distinct p' ∧ path G u p' v"*

Homework 5.1 Split Lists

Submission until Friday, May 18, 11:59am. Recall: Use Isar where appropriate, proofs using *metis*, *smt*, *meson*, *or moura* (as generated by sledgehammer) are forbidden!

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.

lemma
  assumes *"n≥0"* — Note: This assumption is actually not needed, as *n div 0 = 0*, so don't be puzzled if you do not use it at all in your proof.
  shows *"∃ ys zs. length ys = length xs div n ∧ xs=ys@zs"*

Homework 5.2 Estimate Recursion Equation

Submission until Friday, May 18, 11:59am.

(Sledgehammer allowed again)

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

fun *a* :: *"nat ⇒ int"* **where**
"a 0 = 0" |
"a (Suc n) = a n ^ 2 + 1"

We have given you a proof skeleton, setting up the induction. To complete your proof, you should come up with a chain of inequations. You may try to solve the intermediate steps with sledgehammer.

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*]

lemma *"a n ≤ 2 ^ (2 ^ n) − 1"*
proof(*induction n*)
  case *0* **thus** *?case* **by** *simp*
next
  case (*Suc n*)

assume *IH*: *"a n ≤ 2 ˆ 2 ˆ n − 1"*
— Refer to the induction hypothesis by name *IH* or *Suc.IH*
show *"a (Suc n) ≤ 2 ˆ 2 ˆ Suc n − 1"*
proof −

Insert your proof here

qed
qed