Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Mohammad Abdulaziz, Ph.D.

SS 2020
22. 5. 2020

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)
  show ?case
```

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 $2 \hat{} n \leq 2 * 2 \hat{} n$

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
```

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 $sumto\ f\ n = \sum_{i=0\ldots n} f(i)$.
- Show that $(\sum_{i=0\ldots n} i)^2 = \sum_{i=0\ldots n} i^3$.

fun *sumto* :: *"(nat ⇒ nat) ⇒ nat ⇒ nat"*

You may need the following lemma:

lemma *sum_of_naturals*: *"2 * sumto (λx. x) n = n * (n + 1)"*
by (*induction n*) *auto*

lemma *"sumto (λx. x) n ^ 2 = sumto (λx. x^3) n"*
proof (*induct n*)
case *0* **show** *?case* **by** *simp*
next
case (*Suc n*)
assume *IH*: *"(sumto (λx. x) n)² = sumto (λx. x ^ 3) n"*
note [*simp*] = *algebra_simps*

Extend the simpset only in this block

show *"(sumto (λx. x) (Suc n))² = sumto (λx. x ^ 3) (Suc n)"*

Insert a proof here

qed

Exercise 5.3 Enumeration of Trees

Write a function that generates the set of all trees up to a given height. Show that only trees up to the specified height are contained.

fun *enum* :: *"nat ⇒ unit tree set"* **where**
lemma *enum_sound*: *"t ∈ enum n ⟹ height t ≤ n"*

Show that all trees are contained in the generated set. Hint: this requires a case split. Prove it twice, once with apply scripts and the second using Isar.

lemma *enum_complete*: *"height t ≤ n ⟹ t ∈ enum n"*

lemma *enum_complete_isar*: *"height t ≤ n ⟹ t ∈ enum n"*
lemma *enum_correct*: *"enum h = {t. height t ≤ h}"*

Exercise 5.4 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. Simultaneous induction over both trees.

lemma *pretty_unique*: *"pretty t = pretty t' ⟹ 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 Friday, 29 May, 10:00am. 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 *split_lists*:
shows *"∃ ys zs. length ys = length xs div n ∧ xs=ys@zs"*

Homework 5.2 Identity for a Summation

Submission until Friday, 29 May, 10:00am.

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 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*.