

Homework # 14

due Tuesday, December 13

1 Reading

Please read Chapter 29–30 in your textbook.

2 Problems

Please do the following problems:

- Prove Proposition 30.3.10 without using pictures. You will need to write two inductive proofs.
- Exercise 31.2.1. Give subtyping derivations to prove each of the “Yes” answers.

For each problem, I recommend you first think about the problem, and then read the solution. Finally write up your answer.

3 Programming Using F_ω

The typing system behind F_ω allows us to create ADTs for type constructors, not just types. In this Homework, you will implement two different ways of representing lists: church lists (already done previously) or a new way. The new way represents a list as a pair of a natural number (the length) and a function from natural numbers to represent the list elements. Zero is mapped to the first element, one to the second, etc. The elements “past” the length of the list are undefined; you should use “diverge” to avoid returning a misleading value.

You need to define a (proper) type `ADTList` (not a type constructor!) that includes a type constructor as the witness for a record of *polymorphic* list functions (`nil`, `cons`, `isnil`, `head`, `tail`). You next need to define two *values*: `churchList` and `funcList` of the type `ADTList` that package up these two implementations. Finally, you should call the following function on both of these possibilities:

```
/* test code: NB: most types erased */
testList = lambda ltype : ADTList .
  let {List, r} = ltype in
    let map = lambda f.
      fix (lambda m.
        lambda l.
          if r.isnil l
          then r.nil
          else r.cons (f (r.head l))
                    (m (r.tail l))) in
      let l1 = r.cons 1 (r.cons 2 (r.cons 3 (r.nil))) in
      let l2 = map iseven l1 in
      let f2 = lambda b. if b then 0 else 1 in
      r.head (r.tail (map f2 l2));
```

All the type abstractions `lambda X`, type applications `[T]` have been erased. You will need to replace the types, type abstractions and type applications. In particular, you will need to make `map` polymorphic.

4 Piano Tuning and Dead Cows

Please attend Dr. Webber’s talk. What (if anything) does his talk have to do with types, type checking, type soundness or polymorphism?