Problem Set 4, due October 26.

Given a predicate $\alpha(x)$ that specifies some inductively-defined data in the style we have discussed, the predicate specifying lists of such data is

$$\text{List}_\alpha(\ell) \equiv \forall P. (\forall x. \forall \ell. \alpha(x) \rightarrow P(\ell') \rightarrow P(c \ x \ \ell')) \rightarrow P(n) \rightarrow P(\ell)$$

**Exercise 0.** Prove the nil lemma in second-order logic:

$$\forall \alpha. \text{List}_\alpha(n)$$

Extract from the proof a polymorphic representation of nil in System F.

**Exercise 1.** Prove the cons lemma in second-order logic:

$$\forall \alpha. \forall x. \forall \ell. \alpha(x) \rightarrow \text{List}_\alpha(\ell) \rightarrow \text{List}_\alpha(c \ x \ \ell)$$

Extract from the proof a polymorphic representation of cons in System F.

**Exercise 2.** Here is functional programs represented as axioms that work like cons, only it puts (“tacks”) its argument $x$ at the end of a list instead of at the beginning:

$$\forall x. \text{tack} \ x \ n = c \ x \ n$$

$$\forall x. \forall \ell. \text{tack} \ x \ (c \ e \ \ell) = c \ e \ (\text{tack} \ x \ \ell)$$

Prove the tack lemma:

$$\forall \alpha. \forall x. \forall \ell. \alpha(x) \rightarrow \text{List}_\alpha(\ell) \rightarrow \text{List}_\alpha(\text{tack} \ x \ \ell)$$

Extract from the proof a polymorphic representation of tack in System F.

**Exercise 3.** Here is the first-order representation of the functional programming map function as a pair of axioms:

$$\forall f. \text{map} \ f \ n = n$$

$$\forall f. \forall x. \forall \ell. \text{map} \ f \ (c \ x \ \ell) = c \ (f \ x) \ (\text{map} \ f \ \ell)$$

Prove the map theorem in second-order logic:

$$\forall \alpha. \forall \beta. \forall f. \forall \ell. (\forall x. \alpha(x) \rightarrow \beta(f \ x)) \rightarrow \text{List}_\alpha(\ell) \rightarrow \text{List}_\beta(\text{map} \ f \ \ell)$$

Extract from the proof a polymorphic representation of map in System F.
**Exercise 4.** Here is the first-order representation of the functional program that reverses lists, presented axiomatically:

\[
\text{rev } n = n \\
\forall x. \forall \ell. \text{rev } (c \ x \ \ell) = \text{tack } x \ (\text{rev } \ell)
\]

Notice that the tack function \text{tack} is used in the definition. Prove the reverse theorem in second-order logic:

\[
\forall \alpha. \forall \ell. \text{List}_\alpha(\ell) \rightarrow \text{List}_\alpha(\text{rev } \ell)
\]

Extract from the proof a polymorphic representation of \text{map} in System F.

**Exercise 5.** Ackermann’s function is defined by the following recurrence:

\[
\begin{align*}
A(0, n) &= n + 1 \\
A(m + 1, 0) &= A(m, 1) \\
A(m + 1, n + 1) &= A(m, A(m + 1, n))
\end{align*}
\]

Write a System F term, directly, for this function. (I’ve never tried to prove in second order logic that this function terminates, followed by proof extraction.) Try determining the functions \(A(1, n), A(2, n), A(3, n), A(4, n)\)—at some point, these functions are going to seem impossible to write down...