1 Polymorphic lists

\[ t ::= X \mid t \to t \mid \forall X.t \]

\[ \text{Bool} = \text{True of unit} \mid \text{False of unit} \]
\[ \text{Bool} = \forall X.X \to X \to X \]

\[ \text{Nat} = \text{Succ of nat} \mid \text{Zero of unit} \]
\[ \text{Nat} = \forall X.(X \to X) \to X \to X \]

What is the correspondence between the types of the above two? \text{Succ of nat} is of type \text{nat} \to \text{nat} and \text{Zero of unit} is of type \text{nat}. So \((X \to X)\) in the typing for \text{Nat} corresponds to the first \(X\) in the typing for \text{Bool}.

Now let’s give typing for an integer list in System F:

\[ \text{intList} = \text{Cons of int} \ast \text{intList} \mid \text{Nil} \]
\[ \text{CintList} = \forall X.(\text{int} \to X \to X) \to X \to X \]

How about polymorphic lists?

\[ \text{nil} = \lambda X.\lambda f : \text{int} \to X \to X.\lambda z : X.z \]
\[ \text{cons} : \text{int} \to \text{CintList} \to \text{CintList} \]
\[ \text{cons} = \lambda i : \text{int}.\lambda l.\text{CintList}.\lambda X.\lambda f : \text{int} \to X \to X.\lambda z : X.f i(l[X] f z) \]

In the book this is given as

\[ \text{cons} = \lambda h : \text{int}.\lambda l.\text{CintList}.\lambda X.\lambda c : \text{int} \to X \to X.\lambda n : X.\text{cons} \text{hd}(tl[X] \text{ cons} \text{ nil}) \]

What would happen if we wanted to encode function spaces:

\[ \text{val} = \text{Int of int} \mid \text{Fun of val} \to \text{val} \]
\[ \text{Cval} = \forall X.(\text{int} \to X) \to ((X \to X) \to X) \to X \]
\[ \text{int} : \text{int} \to \text{val} \]
\[ \text{int} = \lambda i : \text{int}.\lambda X.\lambda f_{\text{int}} : (\text{int} \to X).\lambda f_{\text{fun}} : (X \to X) \to X.f_{\text{int}} i \]

\[ \text{fun} : (\text{val} \to \text{val}) \to \text{val} \]
\[ \text{fun} : \lambda f(\text{val} \to \text{val}).\lambda X.\lambda f_{\text{int}} : (\text{int} \to X).\lambda f_{\text{fun}} : (X \to X) \to X.\text{SOMETHING}. \]

HW: Write a fold operator in place of SOMETHING.