Recursive Types

int list : nil (of unit)
　　　| cons of int, intlist

$[1;2;3] \triangleq$ cons (1, cons (2, cons (3, nil)))

int list $\triangleq$ unit + int $\times$ int list
　　　　　　　　　　　oops

$\tau ::= \ldots \mid \mu\alpha.\tau$
　　recursive instance of type

remember: fact $\triangleq$ fix $\lambda f. \lambda n. \ldots$
　　recursive instance of fact

int list $\triangleq \mu\alpha.$ unit + int $\times \alpha$

So, does inr (1, nil) : int list?
Not quite.

inr e : ― + ―　　but int list has a μ on the outside
inr (1, nil) : unit + int $\times$ int list $\neq$ int list

$e ::= \ldots \mid \text{fold}_\tau\ e \mid \text{unfold}_\tau\ e$

$$\frac{\Gamma \vdash e : [\mu\alpha.\tau/\alpha]\tau}{\Gamma \vdash \text{fold}_{\mu\alpha.\tau}\ e : \mu\alpha.\tau} \qquad \frac{\Gamma \vdash e : \mu\alpha.\tau}{\Gamma \vdash \text{unfold}_{\mu\alpha.\tau}\ e : [\mu\alpha.\tau/\alpha]\tau}$$

$$\frac{\cdot;\cdot \vdash \mathsf{inl}\ () : \mathsf{unit} + \mathsf{int} \times \mathsf{int\ list}}{\cdot;\cdot \vdash \mathsf{fold}_{\mathsf{int\ list}}\ \mathsf{inl}\ () : \mu\alpha.\ \mathsf{unit} + \mathsf{int} \times \mathsf{int\ list} \triangleq \mathsf{int\ list}}$$

$\mathsf{nil} \triangleq \mathsf{fold}_{\mathsf{int\ list}}\ \mathsf{inl}\ ()$

$\mathsf{cons}\ e_1^{:\mathsf{int}}\ e_2^{:\mathsf{int\ list}} \triangleq \mathsf{fold}_{\mathsf{int\ list}}\ \mathsf{inr}\ (e_1, e_2)$

$\mathsf{hd\_or\_0} : \mathsf{int\ list} \to \mathsf{int}$

$\triangleq \lambda l : \mu\alpha.\ \mathsf{unit} + \mathsf{int} \times \alpha.\ \mathsf{case\ unfold}\ l\ \mathsf{of}$
$\{x.\ 0;$
$y.\ \mathsf{fst}\ y\}$

$$\frac{\Gamma, l : \mu\alpha.\ \mathsf{unit} + \mathsf{int} \times \alpha \vdash l : \mu\alpha.\ \mathsf{unit} + \mathsf{int} \times \alpha}{\Gamma, l : \cdots \vdash \mathsf{unfold}\ l : [\mu\alpha.\ \mathsf{unit} + \mathsf{int} \times \alpha / \alpha](\mathsf{unit} + \mathsf{int} \times \alpha)}$$

$= [\mathsf{int\ list} / \alpha](\mathsf{unit} + \mathsf{int} \times \alpha)$

$= \mathsf{unit} + \mathsf{int} \times \mathsf{int\ list}$

$$\frac{e\ \mathsf{val}}{\mathsf{fold}_\tau\ e\ \mathsf{val}} \qquad \frac{e \mapsto e'}{\mathsf{fold}_\tau\ e \mapsto \mathsf{fold}_\tau\ e'} \qquad \frac{e \mapsto e'}{\mathsf{unfold}\ e \mapsto \mathsf{unfold}\ e'} \qquad \frac{e\ \mathsf{val}}{\mathsf{unfold}(\mathsf{fold}_\tau\ e) \mapsto e}$$

General Recursion

$\mathsf{sum} : \mathsf{int\ list} \to \mathsf{int}$

$\triangleq \lambda l : \mathsf{int\ list}.\ \mathsf{case\ unfold}\ l\ \mathsf{of}$
$\{\_.\ 0;$
$x.\ (\mathsf{fst}\ x) + \mathsf{sum}\ (\mathsf{snd}\ x)\}$

oops
need another fixed pt combinator

let's just add it.

$e ::= \cdots \mid \mathsf{fix}\ x.e$

$$\frac{\Gamma, x : \tau \vdash e : \tau}{\Gamma \vdash \mathsf{fix}\ x.e : \tau} \qquad \overline{\mathsf{fix}\ x.e \mapsto [\mathsf{fix}\ x.e / x]e}$$

$sum \triangleq \text{fix } sum.\ \lambda l : \text{int list}.\ \text{case unfold } l \text{ of}$
$\{\_.\ 0,$
$x.\ (\text{fst } x) + sum(\text{snd } x)\}$

$dbl : \text{int list} \to \text{int list}$
$\triangleq \text{fix } dbl.\ \lambda l : \text{int list}.\ \text{case unfold } l \text{ of}$
$\{\_.\ \text{fold}(\text{inl}\ ())$
$x.\ \text{fold}(\text{inr}\ (2 * \text{fst } x,\ dbl\ (\text{snd } x)))\}$

Can write infinite loops again:

$\text{fix } x.x \mapsto [\text{fix } x.x/x]x = \text{fix } x.x \mapsto \ldots$

In fact, could do it without fix:

$D \triangleq \mu d.\ d \to d$

$$\dfrac{\dfrac{x : D \vdash e : D}{\vdash \lambda x : D.\ e : D \to D}}{\vdash \text{fold}_D(\lambda x : D.\ e) : D}$$

$$\dfrac{\dfrac{x : D \vdash \text{unfold}_D\ x : D \to D \qquad x : D \vdash x : D}{x : D \vdash (\text{unfold}_D\ x)\ x : D}}{\vdash \lambda x : D.(\text{unfold}_D\ x)\ x : D \to D}$$

$\text{fold}_D(\lambda x : D.\ (\text{unfold}_D\ x)\ x) : D$

$(\lambda x : D.(\text{unfold}_D\ x)\ x))\ \text{fold}_D(\lambda x : D.\ (\text{unfold}_D\ x)\ x) : D$

$\approx$

$(\lambda x.\ x\ x)\ (\lambda x. x\ x)$