A Program Fusion Tool

Alberto Pardo
Instituto de Computación
Universidad de la República
Montevideo - Uruguay

joint work with

Facundo Domínguez
Marcelo Gioirgi
Leonardo Richero

Deforestation

$$lenfil\ p = length \circ filter\ p$$

$$\begin{aligned} &length\ [\,] = 0 \\ &length\ (x : xs) = h\ x\ (length\ xs) \\ &\qquad\qquad\qquad\quad \textbf{where} \\ &\qquad\qquad\qquad\qquad h\ x\ n = 1 + n \end{aligned}$$

$$\begin{aligned} &filter\ p\ [\,] = [\,] \\ &filter\ p\ (a : as) = \textbf{if}\ p\ a\ \textbf{then}\ a : filter\ p\ as \\ &\qquad\qquad\qquad\qquad\qquad\quad \textbf{else}\ filter\ p\ as \end{aligned}$$

Deforestation

$$
\begin{aligned}
&\mathit{length}\ [\,] = 0\\
&\mathit{length}\ (x : xs) = h\ x\ (\mathit{length}\ xs)\\
&\qquad\qquad\qquad\quad \textbf{where}\\
&\qquad\qquad\qquad\qquad h\ x\ n = 1 + n
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{filter}\ p\ [\,] = [\,]\\
&\mathit{filter}\ p\ (a : as) = \textbf{if}\ p\ a\ \textbf{then}\ a : \mathit{filter}\ p\ as\\
&\qquad\qquad\qquad\qquad\qquad\quad \textbf{else}\ \mathit{filter}\ p\ as
\end{aligned}
$$

The result:

$$
\begin{aligned}
&\mathit{lenfil}\ p\ [\,] = 0\\
&\mathit{lenfil}\ p\ (a : as) = \textbf{if}\ p\ a\ \textbf{then}\ h\ a\ (\mathit{lenfil}\ p\ as)\\
&\qquad\qquad\qquad\qquad\qquad\quad \textbf{else}\ \mathit{lenfil}\ p\ as\\
&\qquad\qquad\qquad\quad \textbf{where}\\
&\qquad\qquad\qquad\qquad h\ x\ n = 1 + n
\end{aligned}
$$

The approach

The use of standard recursion program schemes:

- Fold
- Unfold
- Hylomorphism

Capturing the structure of functions

Given

$$
\begin{aligned}
&\mathit{fact} :: \mathit{Int} \to \mathit{Int} \\
&\mathit{fact}\ n \mid n < 1 = 1 \\
&\qquad\quad \mid \mathit{otherwise} = n * \mathit{fact}\ (n - 1)
\end{aligned}
$$

we can define,

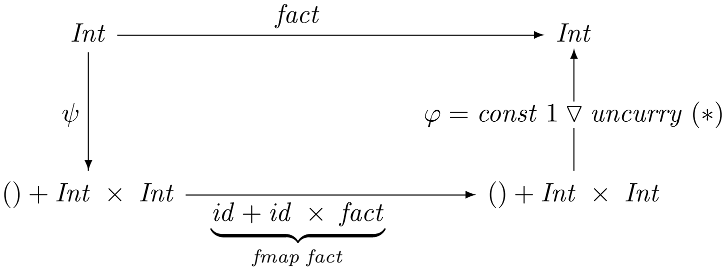
$$
\begin{aligned}
&\psi\ n \mid n < 1 = \mathit{Left}\ () \\
&\qquad \mid \mathit{otherwise} = \mathit{Right}\ (n, n - 1)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{fmap}\ f\ (\mathit{Left}\ ()) = \mathit{Left}\ () \\
&\mathit{fmap}\ f\ (\mathit{Right}\ (m, n)) = \mathit{Right}\ (m, f\ n)
\end{aligned}
$$

$$
\begin{aligned}
&\varphi\ (\mathit{Left}\ ()) = 1 \\
&\varphi\ (\mathit{Right}\ (m, n)) = m * n
\end{aligned}
$$

Capturing the structure of functions (2)

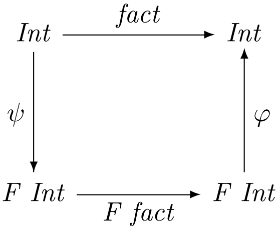
$$fact = \varphi \circ fmap\ fact \circ \psi$$



Capturing the structure of functions (3)

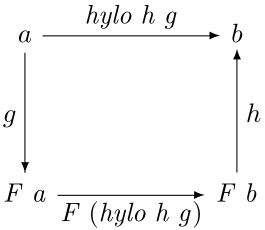
$$F\ a = () + Int\ \times\ a$$

$$F\ f = id + id\ \times\ f$$



Hylomorphism

$$hylo :: (F\ b \to b) \to (a \to F\ a) \to a \to b$$
$$hylo\ h\ g = h \circ hylo\ h\ g \circ g$$



Data types

Functors describe the structure of data types.

Given a data type declaration

$$\mathbf{data}\ \tau = C_1\ \tau_{1,1} \cdots \tau_{1,k_1}\ \mid \cdots \mid\ C_n\ \tau_{n,1} \cdots \tau_{n,k_n}$$

the derivation of the corresp. functor F proceeds as follows:

- pack the arguments to constructors in tuples;
- for constant constructors place the empty tuple ();
- regard alternatives as sums (replace $\mid$ by $+$);
- substitute the occurrences of τ by a type variable a in every $\tau_{i,j}$.

Constructors / Destructors

There exists an isomorphism

$$F\mu F \underset{out_F}{\overset{in_F}{\rightleftarrows}} \mu F$$

where

- in_F encodes the constructors of the data type
- out_F encodes the destructors

Example

Leaf-labelled binary trees

$$\textbf{data}\ Btree\ a = Leaf\ \ a \mid Join\ (Btree\ a)\ (Btree\ a)$$

$$\textbf{type}\ B_a\ b = a + b\ \times\ b$$

$$B_a :: (b \to c) \to (B_a\ b \to B_a\ c)$$
$$B_a\ f = id + f\ \times\ f$$

$$in_{B_a} :: B_a\ (Btree\ a) \to Btree\ a$$
$$in_{B_a} = Leaf \triangledown uncurry\ Join$$

$$out_{B_a} :: Btree\ a \to B_a\ (Btree\ a)$$
$$out_{B_a}\ (Leaf\ \ a) = Left\ a$$
$$out_{B_a}\ (Join\ t\ t') = Right\ (t, t')$$

Fold / Unfold

Fold

$$fold :: (F\ a \to a) \to \mu F \to a$$
$$fold\ h = hylo\ h\ out_F$$

Unfold

$$unfold :: (a \to F\ a) \to a \to \mu F$$
$$unfold\ g = hylo\ in_F\ g$$

Factorisation

$$hylo\ h\ g = fold\ h \circ unfold\ g$$

Fusion laws

Factorisation

$$hylo\ h\ g = hylo\ h\ out_F \circ hylo\ in_F\ g$$

Hylo-Fold Fusion

$$h \text{ strict } \wedge\ \tau :: \forall\ a\ .\ (F\ a \to a) \to (G\ a \to a)$$

$$\Rightarrow$$

$$fold\ h \circ hylo\ (\tau\ in_F)\ g = hylo\ (\tau\ h)\ g$$

Unfold-Hylo Fusion

$$\sigma :: \forall\ a\ .\ (a \to F\ a) \to (a \to G\ a)$$

$$\Rightarrow$$

$$hylo\ h\ (\sigma\ out_F) \circ unfold\ g = hylo\ h\ (\sigma\ g)$$

Factorisation

$$\textbf{data}\ \mathit{Tree}\ a = \mathit{Empty} \mid \mathit{Node}\ (\mathit{Tree}\ a)\ a\ (\mathit{Tree}\ a)$$

$$\begin{aligned}
&\mathit{qsort} :: \mathit{Ord}\ a \Rightarrow [a] \rightarrow [a] \\
&\mathit{qsort} = \mathit{inorder} \circ \mathit{mkTree}
\end{aligned}$$

$$\begin{aligned}
&\mathit{inorder} :: \mathit{Tree}\ a \rightarrow \mathit{List}\ a \\
&\mathit{inorder}\ \mathit{Empty} = \mathit{Nil} \\
&\mathit{inorder}\ (\mathit{Node}\ t\ a\ t') = \mathit{inorder}\ t\ +\!\!+\ [a]\ +\!\!+\ \mathit{inorder}\ t'
\end{aligned}$$

$$\begin{aligned}
&\mathit{mkTree} :: \mathit{Ord}\ a \Rightarrow [a] \rightarrow \mathit{Tree}\ a \\
&\mathit{mkTree}\ [\,] = \mathit{Empty} \\
&\mathit{mkTree}\ (a : as) = \mathit{Node}\ (\mathit{mkTree}\ [x \mid x \leftarrow as; x \leqslant a]) \\
&\qquad\qquad a \\
&\qquad\qquad (\mathit{mkTree}\ [x \mid x \leftarrow as; x > a])
\end{aligned}$$

Hylo-Fold Fusion

$$\textbf{data}\ Maybe\ a = Nothing \mid Just\ a$$

$$mapcoll :: (a \to b) \to List\ (Maybe\ a) \to List\ b$$
$$mapcoll = map\ f \circ collect$$

$$map\ f\ Nil = Nil$$
$$map\ f\ (Cons\ a\ as) = Cons\ (f\ a)\ (map\ f\ as)$$

$$collect :: List\ (Maybe\ Int) \to List\ Int$$
$$collect\ Nil = Nil$$
$$collect\ (Cons\ m\ ms) = \textbf{case}\ m\ \textbf{of}$$
$$\qquad Nothing \to collect\ ms$$
$$\qquad Just\ a \to Cons\ a\ (collect\ ms)$$

Hylo-Fold Fusion

$$
\begin{aligned}
&\tau :: (b, a \to b \to b) \to (b, \mathit{Maybe}\ a \to b \to b) \\
&\tau\ (h_1, h_2) = (h_1, \\
&\qquad\qquad \lambda m\ b \to \mathbf{case}\ m\ \mathbf{of} \\
&\qquad\qquad\qquad \mathit{Nothing} \to b \\
&\qquad\qquad\qquad \mathit{Just}\ a \to h_2\ a\ b)
\end{aligned}
$$

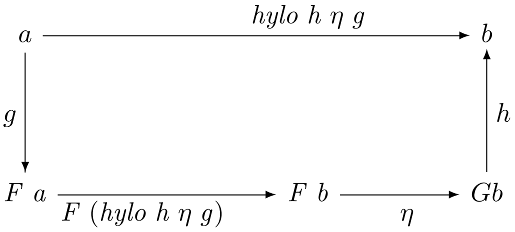
The Tool

- The tool is an extension of the HYLO system presented by Onoue et al. (Univ. of Tokyo).

- Another source was Jacob Schwartz' M.Sc. thesis (MIT, 2000), who implementated the HYLO system in the context of pH (parallel Haskell).

Hylos as triples

$$hylo :: (G\ b \to b) \to (F \Rightarrow G) \to (a \to F\ a) \to a \to b$$
$$hylo\ h\ \eta\ g = h \circ \eta \circ hylo\ h\ \eta\ g \circ g$$



Partial Deforestation

$$\textbf{data}\ \mathit{Btree}\ a = \mathit{Leaf}\ a \mid \mathit{Join}\ (\mathit{Btree}\ a)\ (\mathit{Btree}\ a)$$

$$\mathit{mm}\ f = \mathit{maplBT}\ f \circ \mathit{mirror}$$

$$\begin{aligned}
&\mathit{maplBT} :: (a \to a) \to \mathit{Btree}\ a \to \mathit{Btree}\ a\\
&\mathit{maplBT}\ f\ (\mathit{Leaf}\ a) = \mathit{Leaf}\ (f\ a)\\
&\mathit{maplBT}\ f\ (\mathit{Join}\ t1\ t2) = \mathit{Join}\ (\mathit{maplBT}\ f\ t1)\ t2
\end{aligned}$$

$$\begin{aligned}
&\mathit{mirror} :: \mathit{Btree}\ a \to \mathit{Btree}\ a\\
&\mathit{mirror}\ (\mathit{Leaf}\ a) = \mathit{Leaf}\ a\\
&\mathit{mirror}\ (\mathit{Join}\ t1\ t2) = \mathit{Join}\ (\mathit{mirror}\ t2)\ (\mathit{mirror}\ t1)
\end{aligned}$$

Partial Deforestation

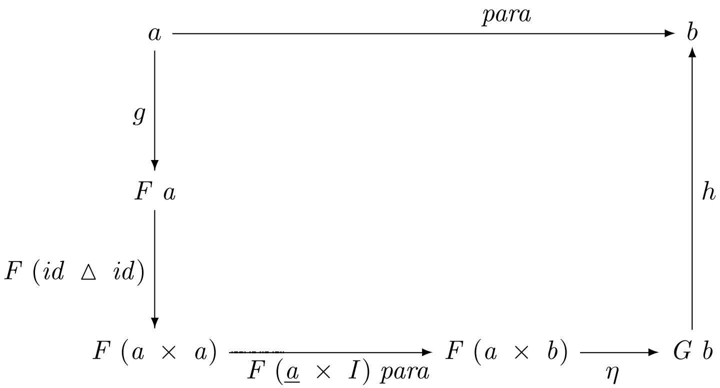
$$mp\ f = maplT\ f \circ prunel$$

$$maplT :: (a \to a) \to Tree\ a \to Tree\ a$$
$$maplT\ f\ Empty = Empty$$
$$maplT\ f\ (Node\ t1\ a\ t2) = Node\ (maplT\ f\ t1)\ (f\ a)\ t2$$

$$prunel :: (a \to Bool) \to Tree\ a \to Tree\ a$$
$$prunel\ p\ Empty = Empty$$
$$prunel\ p\ (Node\ t1\ a\ t2)$$
$$\quad \mid p\ a = prunel\ p\ t2$$
$$\quad \mid otherwise = Node\ (prunel\ p\ t1)\ a\ (prunel\ p\ t2)$$

Paramorphisms

$$para\ h\ \eta\ g = hylo\ h\ \eta\ F\ (id\ \triangle\ id)$$



Fusion in the presence of effects

Programs with effects

$$
\begin{aligned}
&\mathit{lenline} :: \mathit{IO}\ \mathit{Int}\\
&\mathit{lenLine} = \mathbf{do}\ \mathit{xs} \leftarrow \mathit{getLine}\\
&\qquad\qquad\quad\ \ \mathit{return}\ (\mathit{length}\ \mathit{xs})
\end{aligned}
$$

where

$$
\begin{aligned}
&\mathit{getLine} :: \mathit{IO}\ \mathit{String}\\
&\mathit{getLine} = \mathbf{do}\ c \leftarrow \mathit{getChar}\\
&\qquad\qquad\quad\ \ \mathbf{if}\ c == \texttt{'\textbackslash n'}\ \mathbf{then}\ \mathit{return}\ [\,]\\
&\qquad\qquad\qquad\qquad\qquad\quad\ \mathbf{else\ do}\ \mathit{cs} \leftarrow \mathit{getLine}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \ \mathit{return}\ (c : \mathit{cs})
\end{aligned}
$$

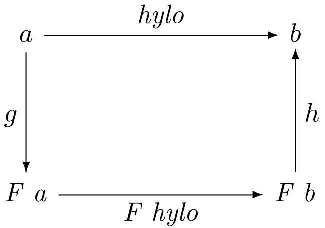
Fusion

$$
\begin{aligned}
&\mathit{length}\ [\,] = 0 \\
&\mathit{length}\ (x : xs) = h\ x\ (\mathit{length}\ xs) \\
&\qquad\qquad\qquad\quad \mathbf{where} \\
&\qquad\qquad\qquad\qquad h\ x\ n = 1 + n
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{getLine} :: \mathit{IO}\ \mathit{String} \\
&\mathit{getLine} = \mathbf{do}\ c \leftarrow \mathit{getChar} \\
&\qquad\qquad\quad \mathbf{if}\ c == \texttt{'\textbackslash n'}\ \mathbf{then}\ \mathit{return}\ [\,] \\
&\qquad\qquad\qquad\qquad\qquad\quad \mathbf{else\ do}\ cs \leftarrow \mathit{getLine} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{return}\ (c : cs)
\end{aligned}
$$

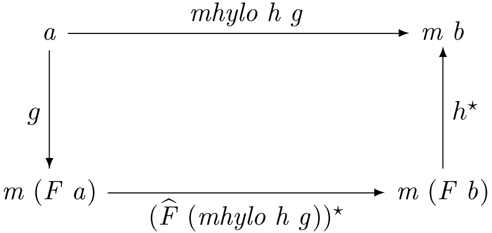
$$
\begin{aligned}
&\mathit{lenLine} = \mathbf{do}\ c \leftarrow \mathit{getChar} \\
&\qquad\qquad\quad \mathbf{if}\ c == \texttt{'\textbackslash n'}\ \mathbf{then}\ \mathit{return}\ 0 \\
&\qquad\qquad\qquad\qquad\qquad\quad \mathbf{else\ do}\ n \leftarrow \mathit{lenLine} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{return}\ (h\ c\ n)
\end{aligned}
$$

Recursion with effects



Monadic hylomorphism

$$mhylo\ h\ g = h \bullet \widehat{F}\ (mhylo\ h\ g) \bullet g$$



$$\widehat{F}\ f \;=\; F\ a \xrightarrow{F\ f} F\ (m\ b) \xrightarrow{dist_F} m\ (F\ b)$$

Monadic hylomorphism

Lists

$$
\begin{aligned}
&\mathit{mhylo}_L :: \mathit{Monad}\ m \Rightarrow \\
&\qquad (m\ c, a \to c \to m\ c) \to (b \to m\ (L_a\ b)) \to (b \to m\ c) \\
&\mathit{mhylo}_L\ (h_1, h_2)\ g \\
&\quad = mh_L \\
&\qquad \mathbf{where} \\
&\qquad\quad mh_L\ b = \mathbf{do}\ x \leftarrow g\ b \\
&\qquad\qquad\qquad\qquad \mathbf{case}\ x\ \mathbf{of} \\
&\qquad\qquad\qquad\qquad\quad \mathit{Left}\ () \to h_1 \\
&\qquad\qquad\qquad\qquad\quad \mathit{Right}\ (a, b') \to \mathbf{do}\ c \leftarrow mh_L\ b' \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad h_2\ a\ c
\end{aligned}
$$

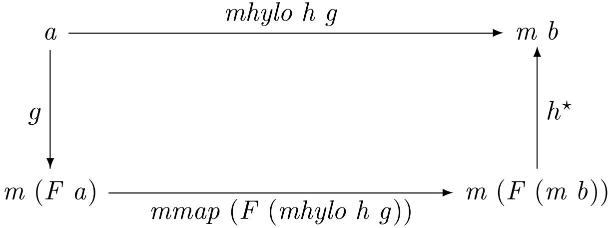
Monadic hylomorphism

Leaf-labelled binary trees

$$
\begin{aligned}
&\mathit{mhylo}_B :: \mathit{Monad}\ m \Rightarrow \\
&\qquad (a \rightarrow m\ c, c \rightarrow c \rightarrow m\ c) \rightarrow \\
&\qquad (b \rightarrow m\ (B_a\ b)) \rightarrow (b \rightarrow m\ c) \\
&\mathit{mhylo}_B\ (h_1, h_2)\ g \\
&\quad = \mathit{mh}_B \\
&\qquad \textbf{where} \\
&\qquad\quad \mathit{mh}_B\ b = \textbf{do}\ x \leftarrow g\ b \\
&\qquad\qquad\qquad\qquad \textbf{case}\ x\ \textbf{of} \\
&\qquad\qquad\qquad\qquad\quad \mathit{Left}\ a \rightarrow h_1\ a \\
&\qquad\qquad\qquad\qquad\quad \mathit{Right}\ (b1, b2) \rightarrow \textbf{do}\ c1 \leftarrow \mathit{mh}_B\ b1 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad c2 \leftarrow \mathit{mh}_B\ b2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad h_2\ c1\ c2
\end{aligned}
$$

A more practical approach

$$mhylo\ h\ g = h\ \bullet\ mmap\ (F\ (mhylo\ h\ g)) \circ g$$

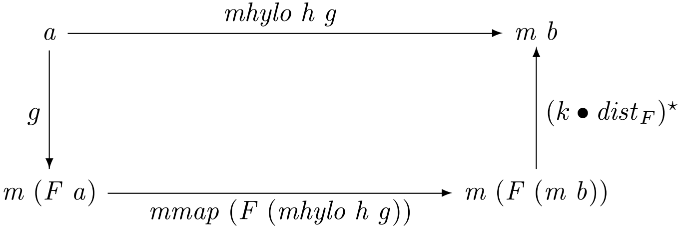


Now $h :: F\ (m\ b) \rightarrow m\ b$ is an algebra with monadic carrier.

A more practical approach

Our previous version of monadic hylomorphism is a particular case.

For $k :: F\ b \to m\ b$,



Examples

$$
\begin{aligned}
&\mathit{sequence} :: \mathit{Monad}\ m \Rightarrow \mathit{List}\ (m\ a) \rightarrow m\ (\mathit{List}\ a)\\
&\mathit{sequence}\ \mathit{Nil} = \mathit{return}\ \mathit{Nil}\\
&\mathit{sequence}\ (\mathit{Cons}\ m\ \mathit{ms}) = \mathbf{do}\ a \leftarrow m\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{as} \leftarrow \mathit{sequence}\ \mathit{ms}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{return}\ (\mathit{Cons}\ a\ \mathit{as})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{msum}_L :: \mathit{Monad}\ m \Rightarrow \mathit{List}\ (m\ \mathit{Int}) \rightarrow m\ \mathit{Int}\\
&\mathit{msum}_L\ \mathit{Nil} = \mathit{return}\ 0\\
&\mathit{msum}_L\ (\mathit{Cons}\ m\ \mathit{ms}) = \mathbf{do}\ x \leftarrow m\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad y \leftarrow \mathit{msum}_L\ \mathit{ms}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{return}\ (x + y)
\end{aligned}
$$

Properties

MHylo-Fold Fusion If $mmap$ is strictness-preserving,

$$h \text{ strict } \wedge \ \tau :: \forall\, a\ .\ (F\ a \rightarrow a) \rightarrow (G\ (m\ a) \rightarrow m\ a)$$

$$\Rightarrow$$

$$mmap\ (fold\ h) \circ mhylo\ (\tau\ in_F)\ g = mhylo\ (\tau\ h)\ g$$

Unfold-MHylo Fusion

$$\sigma :: (a \rightarrow F\ a) \rightarrow (a \rightarrow m\ (G\ a))$$

$$\Rightarrow$$

$$mhylo\ h\ (\sigma\ out_F) \circ unfold\ g = mhylo\ h\ (\sigma\ g)$$

MHylo-Fold Fusion

$$msum_L :: List\ (m\ Int) \rightarrow m\ Int$$
$$msum_L = mmap\ sum_L \circ sequence$$

$$sum_L :: List\ Int \rightarrow Int$$
$$sum_L\ Nil = 0$$
$$sum_L\ (Cons\ a\ as) = a + sum_L\ as$$

$$sequence :: Monad\ m \Rightarrow List\ (m\ a) \rightarrow m\ (List\ a)$$
$$sequence\ Nil = return\ [\,]$$
$$\begin{aligned} sequence\ (Cons\ m\ ms) = \mathbf{do}\ & a \leftarrow m \\ & as \leftarrow sequence\ ms \\ & return\ (Cons\ a\ as) \end{aligned}$$

MHylo-Fold Fusion

$$
\begin{aligned}
&\tau :: (b, Int \to b \to b) \to (m\ b, m\ Int \to m\ b \to m\ b)\\
&\tau\ (h_1, h_2) = (return\ h_1,\\
&\qquad \lambda m\ mb \to \mathbf{do}\ a \leftarrow m\\
&\qquad\qquad b \leftarrow mb\\
&\qquad\qquad return\ (h_2\ a\ b))
\end{aligned}
$$

$$
\begin{aligned}
&msum_L :: Monad\ m \Rightarrow List\ (m\ Int) \to m\ Int\\
&msum_L\ Nil = return\ 0\\
&msum_L\ (Cons\ m\ ms) = \mathbf{do}\ x \leftarrow m\\
&\qquad y \leftarrow msum_L\ ms\\
&\qquad return\ (x + y)
\end{aligned}
$$

Future directions

- Tupling
- Definitions over multiple arguments (e.g. zip).
- Regular data types (e.g. rose trees).

  $\Rightarrow$ the problem is with the recognition of the map function
- Mutual recursion on functions and types
- Investigate other forms of effects