Free Theorems Involving Type Constructor Classes

Janis Voigtländer

Technische Universität Dresden

ICFP'09

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

$$
\begin{aligned}
&\texttt{echo} :: \mathsf{IO}\ ()\\
&\texttt{echo} = \mathbf{do}\ c \leftarrow \texttt{getChar}\\
&\qquad\qquad\quad \texttt{when}\ (c \neq \text{`}{*}\text{'})\ \$\\
&\qquad\qquad\qquad \mathbf{do}\ \texttt{putChar}\ c\\
&\qquad\qquad\qquad\qquad \texttt{echo}
\end{aligned}
$$

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

```
echo :: IO ()
echo = do c ← getChar
          when (c ≠ '*') $
            do putChar c
               echo
```

Example 2:

```
sequence :: Monad m ⇒ [m a] → m [a]
sequence []       = return []
sequence (m : ms) = do a ← m
                       as ← sequence ms
                       return (a : as)
```

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

$$\begin{aligned}&\texttt{echo} :: \mathsf{IO}\ ()\\ &\texttt{echo} = \mathbf{do}\ c \leftarrow \texttt{getChar}\\ &\qquad\qquad\texttt{when}\ (c \neq \text{'}*\text{'})\ \$\\ &\qquad\qquad\ \ \mathbf{do}\ \texttt{putChar}\ c\\ &\qquad\qquad\qquad\ \texttt{echo}\end{aligned}$$

Effectful operations!

Example 2:

$$\begin{aligned}&\texttt{sequence} :: \mathsf{Monad}\ m \Rightarrow [m\ a] \rightarrow m\ [a]\\ &\texttt{sequence}\ [\,] \qquad\quad = \texttt{return}\ [\,]\\ &\texttt{sequence}\ (m : ms) = \mathbf{do}\ a \leftarrow m\\ &\qquad\qquad\qquad\qquad\qquad as \leftarrow \texttt{sequence}\ ms\\ &\qquad\qquad\qquad\qquad\qquad \texttt{return}\ (a : as)\end{aligned}$$

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

A specific monad!

$$
\begin{aligned}
&\texttt{echo} :: \mathrm{IO}\ ()\\
&\texttt{echo} = \mathbf{do}\ c \leftarrow \texttt{getChar}\\
&\qquad\qquad \texttt{when}\ (c \neq \text{`}{*}\text{'})\ \$\\
&\qquad\qquad\quad \mathbf{do}\ \texttt{putChar}\ c\\
&\qquad\qquad\qquad\quad \texttt{echo}
\end{aligned}
$$

Effectful operations!

Example 2:

$$
\begin{aligned}
&\texttt{sequence} :: \mathrm{Monad}\ m \Rightarrow [m\ a] \rightarrow m\ [a]\\
&\texttt{sequence}\ [\,] \qquad\quad = \texttt{return}\ [\,]\\
&\texttt{sequence}\ (m : ms) = \mathbf{do}\ a \leftarrow m\\
&\qquad\qquad\qquad\qquad\qquad as \leftarrow \texttt{sequence}\ ms\\
&\qquad\qquad\qquad\qquad\qquad \texttt{return}\ (a : as)
\end{aligned}
$$

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

A specific monad!

$$
\begin{aligned}
&\texttt{echo} :: \mathsf{IO}\ ()\\
&\texttt{echo} = \textbf{do}\ c \leftarrow \texttt{getChar}\\
&\qquad\qquad\quad \texttt{when}\ (c \neq \text{`}{*}\text{'})\ \$\\
&\qquad\qquad\quad\ \ \textbf{do}\ \texttt{putChar}\ c\\
&\qquad\qquad\qquad\quad \texttt{echo}
\end{aligned}
$$

Effectful operations!

Example 2:

Parametric over a monad!

$$
\begin{aligned}
&\texttt{sequence} :: \mathsf{Monad}\ m \Rightarrow [m\ a] \rightarrow m\ [a]\\
&\texttt{sequence}\ [\,] \qquad\quad\ = \texttt{return}\ [\,]\\
&\texttt{sequence}\ (m : ms) = \textbf{do}\ a \leftarrow m\\
&\qquad\qquad\qquad\qquad\qquad\ as \leftarrow \texttt{sequence}\ ms\\
&\qquad\qquad\qquad\qquad\qquad\ \texttt{return}\ (a : as)
\end{aligned}
$$

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 1:

A specific monad!

```
echo :: IO ()
echo = do c ← getChar
          when (c ≠ '*') $
            do putChar c
               echo
```

Effectful operations!

Example 2:

Parametric over a monad!

```
sequence :: Monad m ⇒ [m a] → m [a]
sequence []       = return []
sequence (m : ms) = do a ← m
                       as ← sequence ms
                       return (a : as)
```

No specific (new) effects!

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 2:

Parametric over a monad!

$$
\begin{aligned}
&\texttt{sequence} :: \text{Monad } m \Rightarrow [m\ a] \to m\ [a] \\
&\texttt{sequence}\ [\,] \qquad\quad\ = \texttt{return}\ [\,] \\
&\texttt{sequence}\ (m : ms) = \textbf{do}\ a \leftarrow m \\
&\qquad\qquad\qquad\qquad\qquad\ \ as \leftarrow \texttt{sequence}\ ms \\
&\qquad\qquad\qquad\qquad\qquad\ \ \texttt{return}\ (a : as)
\end{aligned}
$$

No specific (new) effects!

Monads in Haskell [Wadler '92, Peyton Jones & W. '93]

Example 2:

Parametric over a monad!

$$\texttt{sequence} :: \text{Monad } m \Rightarrow [m\ a] \rightarrow m\ [a]$$

No specific
(new) effects!

A Slightly More Simple Example

```
f :: Monad m ⇒ m a → m a → m a
f m1 m2 =
```

A Slightly More Simple Example

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1$$

A Slightly More Simple Example

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1$$
$$a \leftarrow m_1$$

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\qquad\quad m_2
\end{aligned}
$$

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1\\
&\qquad\qquad\qquad a \leftarrow m_1\\
&\qquad\qquad\qquad m_2\\
&\qquad\qquad\qquad b \leftarrow m_1
\end{aligned}
$$

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\qquad\quad m_2 \\
&\qquad\qquad\qquad\quad b \leftarrow m_1 \\
&\qquad\qquad\qquad\quad c \leftarrow m_2
\end{aligned}
$$

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \mathbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

No effects introduced!

A Slightly More Simple Example

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

No effects introduced!

But m_1, m_2 may encapsulate ones!

A Slightly More Simple Example

Assume m_1, m_2 are pure.

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\qquad\quad m_2 \\
&\qquad\qquad\qquad\quad b \leftarrow m_1 \\
&\qquad\qquad\qquad\quad c \leftarrow m_2 \\
&\qquad\qquad\qquad\quad \texttt{return}\ b
\end{aligned}
$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v. Then:

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ \texttt{return}\ u \\
&\qquad\qquad\qquad a \leftarrow \texttt{return}\ u \\
&\qquad\qquad\qquad \texttt{return}\ v \\
&\qquad\qquad\qquad b \leftarrow \texttt{return}\ u \\
&\qquad\qquad\qquad c \leftarrow \texttt{return}\ v \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ \texttt{return}\ u \\
&\qquad\qquad\qquad a \leftarrow \texttt{return}\ u \\
&\qquad\qquad\qquad \texttt{return}\ v \\
&\qquad\qquad\qquad b \leftarrow \texttt{return}\ u \\
&\qquad\qquad\qquad c \leftarrow \texttt{return}\ v \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

$$(\texttt{return}\ u) \gg m \quad = \quad m$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \mathbf{do}\ \texttt{return}\ u\\
&\qquad\qquad\qquad a \leftarrow \texttt{return}\ u\\
&\qquad\qquad\qquad \texttt{return}\ v\\
&\qquad\qquad\qquad b \leftarrow \texttt{return}\ u\\
&\qquad\qquad\qquad c \leftarrow \texttt{return}\ v\\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

$$(\texttt{return}\ u) \gg m \quad = \quad m$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$
\begin{array}{l}
\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
\texttt{f}\ m_1\ m_2 = \textbf{do} \\
\qquad a \leftarrow \texttt{return}\ u \\
\qquad \texttt{return}\ v \\
\qquad b \leftarrow \texttt{return}\ u \\
\qquad c \leftarrow \texttt{return}\ v \\
\qquad \texttt{return}\ b
\end{array}
$$

$$(\texttt{return}\ u) \gg m \quad = \quad m$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$
\begin{array}{l}
\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
\texttt{f}\ m_1\ m_2 = \textbf{do} \\
\qquad\qquad a \leftarrow \texttt{return}\ u \\
\qquad\qquad \texttt{return}\ v \\
\qquad\qquad b \leftarrow \texttt{return}\ u \\
\qquad\qquad c \leftarrow \texttt{return}\ v \\
\qquad\qquad \texttt{return}\ b
\end{array}
$$

$$(\texttt{return}\ u) \gg\!= (\lambda a \rightarrow m) \quad = \quad m[u/a]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{array}{l} \texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\ \texttt{f}\ m_1\ m_2 = \textbf{do} \end{array}$$

$$\begin{array}{l} \texttt{return}\ v \\ b \leftarrow \texttt{return}\ u \\ c \leftarrow \texttt{return}\ v \\ \texttt{return}\ b \end{array}$$

$$(\texttt{return}\ u) \gg\!= (\lambda a \rightarrow m) \quad = \quad m[u/a]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v. Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}$$

$$\texttt{return}\ v$$
$$b \leftarrow \texttt{return}\ u$$
$$c \leftarrow \texttt{return}\ v$$
$$\texttt{return}\ b$$

$$(\texttt{return}\ v) \gg m \quad = \quad m$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}
\end{aligned}$$

$$\begin{aligned}
&b \leftarrow \texttt{return}\ u \\
&c \leftarrow \texttt{return}\ v \\
&\texttt{return}\ b
\end{aligned}$$

$$(\texttt{return}\ v) \gg m \quad = \quad m$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v. Then:

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}
\end{aligned}
$$

$$
\begin{aligned}
&b \leftarrow \texttt{return}\ u \\
&c \leftarrow \texttt{return}\ v \\
&\texttt{return}\ b
\end{aligned}
$$

$$(\texttt{return}\ u) \gg\!= (\lambda b \rightarrow m) \quad = \quad m[u/b]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{array}{l} \texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\ \texttt{f}\ m_1\ m_2 = \textbf{do} \end{array}$$

$$\begin{array}{l} c \leftarrow \texttt{return}\ v \\ \texttt{return}\ u \end{array}$$

$$(\texttt{return}\ u) \gg\!= (\lambda b \rightarrow m) \quad = \quad m[u/b]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{array}{l} \texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\ \texttt{f}\ m_1\ m_2 = \textbf{do} \end{array}$$

$$\begin{array}{l} c \leftarrow \texttt{return}\ v \\ \texttt{return}\ u \end{array}$$

$$(\texttt{return}\ v) \gg\!= (\lambda c \rightarrow m) \quad = \quad m[v/c]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{aligned}&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a\\&\texttt{f}\ m_1\ m_2 = \textbf{do}\end{aligned}$$

$$\texttt{return}\ u$$

$$(\texttt{return}\ v) \gg\!= (\lambda c \to m) \quad = \quad m[v/c]$$

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\begin{array}{l} \texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a \\ \texttt{f}\ m_1\ m_2 = \textbf{do} \end{array}$$

$$\texttt{return}\ u$$

Purity is propagated!

A Slightly More Simple Example

Assume m_1, m_2 are pure.

That is, $m_1 = (\texttt{return}\ u)$ and $m_2 = (\texttt{return}\ v)$ for some u, v.
Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}$$

$\texttt{return}\ u$

Purity is propagated!

What about other “invariants”?

Propagating Invariants

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State $\sigma\ \tau$,

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\quad m_2 \\
&\qquad\qquad\quad b \leftarrow m_1 \\
&\qquad\qquad\quad c \leftarrow m_2 \\
&\qquad\qquad\quad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\quad m_2 \\
&\qquad\qquad\quad b \leftarrow m_1 \\
&\qquad\qquad\quad c \leftarrow m_2 \\
&\qquad\qquad\quad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

Can we show that $\texttt{execState}\ (\texttt{f}\ m_1\ m_2) = \texttt{id}$?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` ($\texttt{f}\ m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}^{s}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}\,m_1{}^{s} \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \mathbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad\qquad\quad a \leftarrow {}^{s}m_1 \\
&\qquad\qquad\quad m_2 \\
&\qquad\qquad\quad b \leftarrow m_1 \\
&\qquad\qquad\quad c \leftarrow m_2 \\
&\qquad\qquad\quad \texttt{return}\ b
\end{aligned}$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad {}^{s}m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad {}^{s}m_2{}^{s} \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\mathtt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\mathtt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad {}^{s}m_2{}^{s} \\
&\qquad\qquad\qquad b \leftarrow {}^{s}m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \mathtt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i = $ `id`.

Can we show that `execState` $(\texttt{f}\ m_1\ m_2) = $ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s}\\
&\qquad a \leftarrow {}^{s}m_1{}^{s}\\
&\qquad {}^{s}m_2{}^{s}\\
&\qquad b \leftarrow {}^{s}m_1{}^{s}\\
&\qquad c \leftarrow m_2\\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad\quad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad\quad {}^{s}m_2{}^{s} \\
&\qquad\qquad\qquad\quad b \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad\quad c \leftarrow {}^{s}m_2 \\
&\qquad\qquad\qquad\quad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \mathrm{State}\ \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \mathrm{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \mathbf{do}\ {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad a \leftarrow {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad {}^{s}m_2{}^{s}\\
&\qquad\qquad\qquad b \leftarrow {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad c \leftarrow {}^{s}m_2{}^{s}\\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad a \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad {}^{s}m_2{}^{s} \\
&\qquad\qquad\qquad b \leftarrow {}^{s}m_1{}^{s} \\
&\qquad\qquad\qquad c \leftarrow {}^{s}m_2{}^{s} \\
&\qquad\qquad\qquad {}^{s}\texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State $\sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad a \leftarrow {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad {}^{s}m_2{}^{s}\\
&\qquad\qquad\qquad b \leftarrow {}^{s}m_1{}^{s}\\
&\qquad\qquad\qquad c \leftarrow {}^{s}m_2{}^{s}\\
&\qquad\qquad\qquad {}^{s}\texttt{return}\ b^{s}
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda c \rightarrow \text{State}\ (\lambda s \rightarrow (b, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

Can we show that $\texttt{execState}\ (\texttt{f}\ m_1\ m_2) = \texttt{id}$?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda c \rightarrow \text{State}\ (\lambda s \rightarrow (b, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` m_i = `id`.

Can we show that `execState` (`f` $m_1\ m_2$) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

Can we show that `execState` (`f` m_1 m_2) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow \text{State } (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State } (\lambda s \rightarrow (b, s))
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State σ τ, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` m_1 m_2) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (b, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda b \rightarrow \text{State}\ (\lambda s \rightarrow (b, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda b \rightarrow \text{State}\ (\lambda s \rightarrow (b, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad a \leftarrow m_1 \\
&\qquad\qquad\qquad\quad m_2 \\
&\qquad\qquad\qquad\quad \text{State } (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

Can we show that $\texttt{execState}\ (\texttt{f}\ m_1\ m_2) = \texttt{id}$?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1\\
&\qquad\qquad\quad a \leftarrow m_1\\
&\qquad\qquad\quad \text{State}\ (\lambda s \rightarrow (\cdots, s))\\
&\qquad\qquad\quad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg (\text{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a\\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1\\
&\qquad\qquad\qquad a \leftarrow m_1\\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg (\text{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

Can we show that `execState` (`f` m_1 m_2) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow \text{State } (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State } (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda a \rightarrow \text{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

Can we show that $\texttt{execState}\ (\texttt{f}\ m_1\ m_2) = \texttt{id}$?

$$\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg\!= (\lambda a \rightarrow \text{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad\quad \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

Can we show that `execState` (`f` m_1 m_2) = `id`?

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ \text{State}\ (\lambda s \rightarrow (\cdots, s)) \\
&\qquad\qquad\quad\ \ \text{State}\ (\lambda s \rightarrow (\cdots, s))
\end{aligned}
$$

Propagating Invariants

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

Can we show that `execState` (`f` m_1 m_2) = `id`?

$$\mathtt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\mathtt{f}\ m_1\ m_2 = \textbf{do}\ \text{State}\ (\lambda s \rightarrow (\cdots, s))$$
$$\qquad\qquad\qquad \text{State}\ (\lambda s \rightarrow (\cdots, s))$$

$$(\text{State}\ (\lambda s \rightarrow (\cdots, s))) \gg (\text{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \mathsf{State}\ \sigma\ \tau$, but $\mathtt{execState}\ m_i = \mathtt{id}$.

Can we show that $\mathtt{execState}\ (\mathtt{f}\ m_1\ m_2) = \mathtt{id}$?

$$\mathtt{f} :: \mathsf{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\mathtt{f}\ m_1\ m_2 = \mathbf{do}\ \mathsf{State}\ (\lambda s \rightarrow (\cdots, s))$$

$$(\mathsf{State}\ (\lambda s \rightarrow (\cdots, s))) \gg (\mathsf{State}\ (\lambda s \rightarrow (\cdots, s))) \quad = \quad ?$$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` $($`f` $m_1\ m_2) =$ `id`?

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}\ \text{State}\ (\lambda s \rightarrow (\cdots, s))$$

Yes!

Why So?

Crucially used:

- for every a,

$$\texttt{execState}\ (\texttt{return}\ a) = \texttt{id}$$

Why So?

Crucially used:

- for every a,

$$\texttt{execState}\ (\texttt{return}\ a) = \texttt{id}$$

- for every m and k,

$$\texttt{execState}\ (m \gg\!= k) = \texttt{id}$$

  provided:

  - $\texttt{execState}\ m = \texttt{id}$
  - for every a, $\texttt{execState}\ (k\ a) = \texttt{id}$

Propagating Invariants

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

Can we show that `execState` (`f` $m_1\ m_2$) $=$ `id`?

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}\ \text{State}\ (\lambda s \rightarrow (\cdots, s))$$

Yes!

Propagating Invariants

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

Can we show that $\texttt{execState}\ (\texttt{f}\ m_1\ m_2) = \texttt{id}$?

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$
$$\texttt{f}\ m_1\ m_2 = \textbf{do}\ \text{State}\ (\lambda s \rightarrow (\cdots, s))$$

Yes!

What about other invariants, other monads, ...?

Consider a More Specific Type

Instead of

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

now

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \to m\ \text{Int} \to m\ \text{Int}$$

Consider a More Specific Type

Instead of

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

now

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \to m\ \text{Int} \to m\ \text{Int}$$

Then more possible behaviours of `f` are possible:

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \to m\ \text{Int} \to m\ \text{Int} \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad\qquad\qquad a \leftarrow m_1 \\
&\qquad\qquad\qquad m_2 \\
&\qquad\qquad\qquad b \leftarrow m_1 \\
&\qquad\qquad\qquad c \leftarrow m_2 \\
&\qquad\qquad\qquad \texttt{return}\ b
\end{aligned}
$$

Consider a More Specific Type

Instead of

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

now

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \to m\ \text{Int} \to m\ \text{Int}$$

Then more possible behaviours of `f` are possible:

```
f :: Monad m ⇒ m Int → m Int → m Int
f m1 m2 = do m1
             a ← m1
             m2
             b ← m1
             if b > 0 then return (a + b)
                      else do c ← m2
                              return b
```

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but `execState` $m_i =$ `id`.

$$
\begin{aligned}
&\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a \\
&\texttt{f}\ m_1\ m_2 = \textbf{do}\ m_1 \\
&\qquad a \leftarrow m_1 \\
&\qquad m_2 \\
&\qquad b \leftarrow m_1 \\
&\qquad c \leftarrow m_2 \\
&\qquad \texttt{return}\ b
\end{aligned}
$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned} \texttt{f}\ m_1\ m_2 = \textbf{do}\ & m_1 \\ & a \leftarrow m_1 \\ & m_2 \\ & b \leftarrow m_1 \\ & c \leftarrow m_2 \\ & \texttt{return}\ b \end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \to \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned} \texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & \texttt{return}\ b \end{aligned}$$

$$\begin{aligned} \texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b \end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$\texttt{return}\ b \quad = \quad h\ (\texttt{return}\ b)$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \to \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & h\ (\texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader } \sigma\ a \to \text{State } \sigma\ a$$
$$h\ (\text{Reader } g) = \text{State } (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned} \texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & h\ (\texttt{return}\ b) \end{aligned}$$

$$\begin{aligned} \texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b \end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but `execState` $m_i =$ `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & h\ (\texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$(h\ m_2') \gg\!= (\lambda c \rightarrow h\ (\texttt{return}\ b)) \quad = \quad ?$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \to \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & c \leftarrow h\ m_2' \\ & h\ (\texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$(h\ m_2') \gg\!= (\lambda c \to h\ (\texttt{return}\ b)) \quad = \quad h\ (m_2' \gg\!= (\lambda c \to \texttt{return}\ b))$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \to \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \mathbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & h\ (\mathbf{do}\ c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \mathbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$(h\ m_2') \gg\!= (\lambda c \to h\ (\texttt{return}\ b)) \quad = \quad h\ (m_2' \gg\!= (\lambda c \to \texttt{return}\ b))$$

Reasoning via Monad Embedding

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned} \texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & h\ (\textbf{do}\ c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b) \end{aligned}$$

$$\begin{aligned} \texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b \end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & h\ (\textbf{do}\ c \leftarrow m_2' \\ & \qquad\ \ \texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned} \texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & h\ (\textbf{do}\ c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b) \end{aligned}$$

$$\begin{aligned} \texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b \end{aligned}$$

$$(h\ m) \mathbin{>\!\!>\!\!=} (h \circ k) \quad = \quad ?$$

Reasoning via Monad Embedding

Assume m_1, m_2 :: State $\sigma\ \tau$, but `execState` m_i = `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & b \leftarrow h\ m_1' \\ & h\ (\textbf{do}\ c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$(h\ m) \gg\!= (h \circ k) \quad = \quad h\ (m \gg\!= k)$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & a \leftarrow h\ m_1' \\ & h\ m_2' \\ & h\ (\textbf{do}\ b \leftarrow m_1' \\ & \qquad\quad c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b)\end{aligned}
\qquad
\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

$$(h\ m) \gg\!= (h \circ k) \quad = \quad h\ (m \gg\!= k)$$

Reasoning via Monad Embedding

Assume m_1, m_2 :: State σ τ, but `execState` m_i = `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ h\ m_1' ;\ a \leftarrow h\ m_1' ;\ h\ m_2' ;\ h\ (\textbf{do}\ b \leftarrow m_1' ;\ c \leftarrow m_2' ;\ \texttt{return}\ b)$$

$$\texttt{f}\ m_1'\ m_2' = \textbf{do}\ m_1' ;\ a \leftarrow m_1' ;\ m_2' ;\ b \leftarrow m_1' ;\ c \leftarrow m_2' ;\ \texttt{return}\ b$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m'_1)\ (h\ m'_2) = \textbf{do}\ & h\ m'_1 \\ & a \leftarrow h\ m'_1 \\ & h\ (\textbf{do}\ m'_2 \\ & \qquad\quad b \leftarrow m'_1 \\ & \qquad\quad c \leftarrow m'_2 \\ & \qquad\quad \texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m'_1\ m'_2 = \textbf{do}\ & m'_1 \\ & a \leftarrow m'_1 \\ & m'_2 \\ & b \leftarrow m'_1 \\ & c \leftarrow m'_2 \\ & \texttt{return}\ b\end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but `execState` $m_i =$ `id`.

An m has this property iff it is an h-image for

$$h :: \text{Reader } \sigma\ a \to \text{State } \sigma\ a$$
$$h\ (\text{Reader } g) = \text{State } (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\begin{aligned}\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ & h\ m_1' \\ & h\ (\textbf{do}\ a \leftarrow m_1' \\ & \qquad\quad m_2' \\ & \qquad\quad b \leftarrow m_1' \\ & \qquad\quad c \leftarrow m_2' \\ & \qquad\quad \texttt{return}\ b)\end{aligned}$$

$$\begin{aligned}\texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b\end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State}\ \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad}\ m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

$$\begin{aligned} \texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ h\ (\textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b) \end{aligned}$$

$$\begin{aligned} \texttt{f}\ m_1'\ m_2' = \textbf{do}\ & m_1' \\ & a \leftarrow m_1' \\ & m_2' \\ & b \leftarrow m_1' \\ & c \leftarrow m_2' \\ & \texttt{return}\ b \end{aligned}$$

Reasoning via Monad Embedding

Assume $m_1, m_2 :: \text{State } \sigma\ \tau$, but $\texttt{execState}\ m_i = \texttt{id}$.

An m has this property iff it is an h-image for

$$h :: \text{Reader } \sigma\ a \to \text{State } \sigma\ a$$
$$h\ (\text{Reader } g) = \text{State } (\lambda s \to (g\ s, s))$$

Then:

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \to m\ a \to m\ a$$

$$\texttt{f}\ (h\ m_1')\ (h\ m_2') = \textbf{do}\ h\ (\textbf{do}\ \begin{array}{l} m_1' \\ a \leftarrow m_1' \\ m_2' \\ b \leftarrow m_1' \\ c \leftarrow m_2' \\ \texttt{return}\ b) \end{array}$$

$$\texttt{f}\ m_1'\ m_2' = \textbf{do}\ \begin{array}{l} m_1' \\ a \leftarrow m_1' \\ m_2' \\ b \leftarrow m_1' \\ c \leftarrow m_2' \\ \texttt{return}\ b \end{array}$$

$$\texttt{f}\ (h\ m_1')\ (h\ m_2') \ = \ h\ (\texttt{f}\ m_1'\ m_2')$$

A More General Theorem

Let

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

Let

$$h :: \kappa_1\ a \rightarrow \kappa_2\ a$$

such that

- κ_1, κ_2 are monads
- $h \circ \texttt{return}_{\kappa_1} = \texttt{return}_{\kappa_2}$
- for every m and k, $h\ (m \gg=_{\kappa_1} k) = (h\ m) \gg=_{\kappa_2} (h \circ k)$

Then for every m_1 and m_2,

$$\texttt{f}\ (h\ m_1)\ (h\ m_2) \quad = \quad h\ (\texttt{f}\ m_1\ m_2)$$

A More General Theorem

Let

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

Let

$$h :: \kappa_1\ a \rightarrow \kappa_2\ a$$

such that

- κ_1, κ_2 are monads
- $h \circ \texttt{return}_{\kappa_1} = \texttt{return}_{\kappa_2}$
- for every m and k, $h\ (m \gg=_{\kappa_1} k) = (h\ m) \gg=_{\kappa_2} (h \circ k)$

Then for every m_1 and m_2,

$$\texttt{f}\ (h\ m_1)\ (h\ m_2) \quad = \quad h\ (\texttt{f}\ m_1\ m_2)$$

The same for

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \rightarrow m\ \text{Int} \rightarrow m\ \text{Int}$$

A More General Theorem

Let

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

Let

$$h :: \kappa_1\ a \rightarrow \kappa_2\ a$$

such that

- κ_1, κ_2 are monads
- $h \circ \texttt{return}_{\kappa_1} = \texttt{return}_{\kappa_2}$
- for every m and k, $h\ (m \gg\!=_{\kappa_1} k) = (h\ m) \gg\!=_{\kappa_2} (h \circ k)$

Then for every m_1 and m_2,

$$\texttt{f}\ (h\ m_1)\ (h\ m_2) \quad = \quad h\ (\texttt{f}\ m_1\ m_2)$$

The same for

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \rightarrow m\ \text{Int} \rightarrow m\ \text{Int}$$

Looking Back at the Concrete Invariant

For

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

Looking Back at the Concrete Invariant

For

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

the conditions

- $h \circ \texttt{return}_{\text{Reader}\,\sigma} = \texttt{return}_{\text{State}\,\sigma}$
- for every m and k,
  $h\ (m \ggg\!=_{\text{Reader}\,\sigma} k) = (h\ m) \ggg\!=_{\text{State}\,\sigma} (h \circ k)$

Looking Back at the Concrete Invariant

For

$$h :: \text{Reader}\ \sigma\ a \rightarrow \text{State}\ \sigma\ a$$
$$h\ (\text{Reader}\ g) = \text{State}\ (\lambda s \rightarrow (g\ s, s))$$

the conditions

- $h \circ \texttt{return}_{\text{Reader}\,\sigma} = \texttt{return}_{\text{State}\,\sigma}$
- for every m and k,
  $h\ (m \gg\!= _{\text{Reader}\,\sigma}\ k) = (h\ m) \gg\!= _{\text{State}\,\sigma} (h \circ k)$

imply that

- for every a, $\texttt{execState}\ (\texttt{return}_{\text{State}\,\sigma}\ a) = \texttt{id}$
- for every m and k, $\texttt{execState}\ (m \gg\!= _{\text{State}\,\sigma}\ k) = \texttt{id}$,
  provided ... (as given earlier)

A More General Theorem

Let

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ a \rightarrow m\ a \rightarrow m\ a$$

Let

$$h :: \kappa_1\ a \rightarrow \kappa_2\ a$$

such that

- κ_1, κ_2 are monads
- $h \circ \texttt{return}_{\kappa_1} = \texttt{return}_{\kappa_2}$
- for every m and k, $h\ (m \gg\!=_{\kappa_1} k) = (h\ m) \gg\!=_{\kappa_2} (h \circ k)$

Then for every m_1 and m_2,

$$\texttt{f}\ (h\ m_1)\ (h\ m_2) \quad = \quad h\ (\texttt{f}\ m_1\ m_2)$$

The same for

$$\texttt{f} :: \text{Monad } m \Rightarrow m\ \text{Int} \rightarrow m\ \text{Int} \rightarrow m\ \text{Int}$$

Conceptual Ingredients

- Exploiting polymorphism
  - Relational parametricity [Reynolds '83]
  - Free theorems [Wadler '89]

Conceptual Ingredients

- Exploiting polymorphism
  - Relational parametricity [Reynolds '83]
  - Free theorems [Wadler '89]

- Extension to type classes:
  - Folklore
  - Dictionary translation [Wadler & Blott '89]

Conceptual Ingredients

- Exploiting polymorphism
  - Relational parametricity [Reynolds '83]
  - Free theorems [Wadler '89]

- Extension to type classes:
  - Folklore
  - Dictionary translation [Wadler & Blott '89]

- Extension to type constructors:
  - Folklore?

Conceptual Ingredients

- Exploiting polymorphism
  - Relational parametricity [Reynolds '83]
  - Free theorems [Wadler '89]
- Extension to type classes:
  - Folklore
  - Dictionary translation [Wadler & Blott '89]
- Extension to type constructors:
  - Folklore?
- Monad morphisms:
  - Representation independence for effects [Filinski & Støvring '07]

Example Uses

- Purity preservation

Example Uses

- Purity preservation
- Safe value extraction, e.g.
  - Discard logging
  - Pick from a nondeterministic manifold

Example Uses

- Purity preservation
- Safe value extraction, e.g.
  - Discard logging
  - Pick from a nondeterministic manifold
- Invariant propagation, e.g.
  - Independence criteria for stateful computations
  - Restrictions on IO

Example Uses

- Purity preservation
- Safe value extraction, e.g.
  - Discard logging
  - Pick from a nondeterministic manifold
- Invariant propagation, e.g.
  - Independence criteria for stateful computations
  - Restrictions on IO
- Effect abstraction, e.g.
  - From exceptions to partiality

Example Uses

- Purity preservation
- Safe value extraction, e.g.
  - Discard logging
  - Pick from a nondeterministic manifold
- Invariant propagation, e.g.
  - Independence criteria for stateful computations
  - Restrictions on IO
- Effect abstraction, e.g.
  - From exceptions to partiality
- Proper generalisations of standard free theorems

Example Uses

- Purity preservation
- Safe value extraction, e.g.
  - Discard logging
  - Pick from a nondeterministic manifold
- Invariant propagation, e.g.
  - Independence criteria for stateful computations
  - Restrictions on IO
- Effect abstraction, e.g.
  - From exceptions to partiality
- Proper generalisations of standard free theorems
- Transparent introduction of data improvements [V. '08]

References I

A. Filinski and K. Støvring.
Inductive reasoning about effectful data types.
In *International Conference on Functional Programming, Proceedings*, pages 97–110. ACM Press, 2007.

E. Moggi.
Notions of computation and monads.
Information and Computation, 93(1):55–92, 1991.

S.L. Peyton Jones and P. Wadler.
Imperative functional programming.
In *Principles of Programming Languages, Proceedings*, pages 71–84. ACM Press, 1993.

References II

J.C. Reynolds.
Types, abstraction and parametric polymorphism.
In *Information Processing, Proceedings*, pages 513–523. Elsevier, 1983.

J. Voigtländer.
Asymptotic improvement of computations over free monads.
In *Mathematics of Program Construction, Proceedings*, volume 5133 of *LNCS*, pages 388–403. Springer-Verlag, 2008.

P. Wadler.
Theorems for free!
In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 347–359. ACM Press, 1989.

References III

P. Wadler.
The essence of functional programming (Invited talk).
In *Principles of Programming Languages, Proceedings*, pages 1–14. ACM Press, 1992.

P. Wadler and S. Blott.
How to make ad-hoc polymorphism less ad hoc.
In *Principles of Programming Languages, Proceedings*, pages 60–76. ACM Press, 1989.