Composition of Monads

Jeremy E. Dawson

Logic & Computation Programme  Automated Reasoning Group
National ICT Australia  Computer Sciences Laboratory
Res. Sch. of Inf. Sci. and Eng.
Australian National University

Types of monad functions

\[ \text{unit} : \alpha \rightarrow \alpha M \]

\[ \text{map} : (\alpha \rightarrow \beta) \rightarrow (\alpha M \rightarrow \beta M) \]

\[ \text{join} : \alpha MM \rightarrow \alpha M \]

\[ \text{ext} : (\alpha \rightarrow \beta M) \rightarrow (\alpha M \rightarrow \beta M) \]

\[ \text{bind} : \alpha M \rightarrow (\alpha \rightarrow \beta M) \rightarrow \beta M \]

\[ \otimes : (\beta \rightarrow \gamma M) \rightarrow (\alpha \rightarrow \beta M) \rightarrow (\alpha \rightarrow \gamma M) \]

Examples for the list monad

let \( g \ 1 = [1], \ g \ 3 = [1,2,3], \) etc

\[ \text{unit} \ a = [a] \]

\[ \text{map} \ f \ [x,y,z] = [fx, fy, fz] \]

\[ \text{join} \ [[u,v], [w], [x,y]] = [u, v, w, x, y] \]

\[ \text{ext} \ g \ [2,0,4] = [1,2,1,2,3,4] \]
Relationships between monad functions

\[ m \text{ bind } f = \text{ ext } f \ m \]
\[ \text{ext } f = \text{ join } \circ \text{ map } f \]
\[ \text{join} = \text{ ext } \text{id} \]
\[ \text{map } f = \text{ ext } (\text{unit } \circ f) \]
\[ g \odot f = \text{ ext } g \odot f \]
\[ \text{ext } g = g \odot \text{id} \]
Monad rules for *unit*, *map* and *join*

\[
\begin{align*}
\text{map id} &= \text{id} \\
\text{map } f \circ \text{map } g &= \text{map } (f \circ g) \\
\text{unit } f &= \text{map } f \circ \text{unit} \\
\text{join } \circ \text{map } (\text{map } f) &= \text{map } f \circ \text{join} \\
\text{join } \circ \text{unit} &= \text{id} \\
\text{join } \circ \text{map unit} &= \text{id} \\
\text{join } \circ \text{map join} &= \text{join } \circ \text{join} \\
\text{ext } f &= \text{join } \circ \text{map } f \\
\text{ext } (g \circ f) &= \text{ext } g \circ \text{map } f
\end{align*}
\]
Examples of monad rules

\[ \text{join} \circ \text{unit} = \text{id} \]  \hspace{1cm} (5)
\[ \text{join} \left[\left[3, 5, 7\right]\right] = \left[3, 5, 7\right] \]

\[ \text{join} \circ \text{map unit} = \text{id} \]  \hspace{1cm} (6)
\[ \text{join} \left[\left[3\right], \left[5\right], \left[7\right]\right] = \left[3, 5, 7\right] \]

\[ \text{join} \circ \text{map (map f)} = \text{map f} \circ \text{join} \]  \hspace{1cm} (4)
\[ \left[\left[u, v\right], \left[w\right], \left[x, y\right]\right] \quad \left[\left[u', v'\right], \left[w'\right], \left[x', y'\right]\right] \quad \left[u, v, w, x, y\right] \quad \left[u', v', w', x', y'\right] \]

\[ \text{join} \circ \text{map join} = \text{join} \circ \text{join} \]  \hspace{1cm} (7)
\[ \left[\left[\left[u, v\right], \left[w\right]\right], \left[\left[x\right], \left[y, z\right]\right]\right] \quad \left[\left[\left[u, v, w\right], \left[x, y, z\right]\right], \left[u, v, w, x, y, z\right]\right] \]
Monad rules for \textit{unit}, \textit{ext} and $\odot$

\begin{align*}
\text{ext } f \circ \text{unit} & = f \\
\text{ext } \text{unit} & = \text{id} \\
\text{ext } (\text{ext } g \circ f) & = \text{ext } g \circ \text{ext } f \\
\text{join} & = \text{ext } \text{id} \\
\text{map } f & = \text{ext } (\text{unit } \circ f)
\end{align*}

\begin{align*}
\text{ext } f \circ \text{unit} & = f \\
\text{ext } \text{unit} & = \text{id} \\
\text{ext } (g \odot f) & = \text{ext } g \circ \text{ext } f \\
g \odot f & = \text{ext } g \circ f
\end{align*}
A useful theorem

**Theorem 1**  *In a monad the following are equivalent*

(i)  \( \text{ext } g = g \circ \text{join} \)

(ii)  \( g = \text{ext } (g \circ \text{unit}) \)

(iii)  *there exists* \( f \) *such that* \( g = \text{ext } f \)

(iv)  *for all* \( h \),  \( \text{ext } (g \circ h) = g \circ \text{ext } h \)

Refer monad rules (E4), (7), (E5) and (4)
Identity and associativity in the Kleisli category $\mathcal{K}_M$

**Theorem 2** Assuming rules (E1) to (E3)

\[
\begin{align*}
g \circ f &= \text{ext } g \circ f \tag{E6} \\
(h \circ g) \circ f &= h \circ (g \circ f) \tag{A6} \\
\text{ext } f &= \text{ext } g \Rightarrow f = g \tag{E1} \\
f \circ \text{unit} &= f \tag{A1} \\
\text{unit} \circ f &= f \tag{A2} \\
h \circ (g \circ f) &= (h \circ g) \circ f \tag{A3}
\end{align*}
\]

the Kleisli category $\mathcal{K}_M$: **objects** are types — $\alpha$, $\beta$, etc

**an arrow** from $\alpha$ to $\beta$ is a function of type $\alpha \to \beta M$

**identity arrow** on $\alpha$ is $\text{unit} : \alpha \to \alpha M$

**composition** is $\circ : (\beta \to \gamma M) \to (\alpha \to \beta M) \to (\alpha \to \gamma M)$, so $g$ from $\beta$ to $\gamma$ and $f$ from $\alpha$ to $\beta$ compose to give $g \circ f$, from $\alpha$ to $\gamma$

Rules (A1) to (A3) give the properties required for a category.
Monad Rules Based on the Kleisli Category $\mathcal{K}_M$

\[ f \circ \text{unit} = f \]  
\[ \text{unit} \circ f = f \]  
\[ h \circ (g \circ f) = (h \circ g) \circ f \]  
\[ (h \circ \text{id}) \circ f = h \circ f \]  
\[ h \circ (\text{unit} \circ f) = h \circ f \]  
\[ \text{ext} \ g = g \circ \text{id} \]  
\[ (h \circ g) \circ f = h \circ (g \circ f) \]
The State Monad

Let State be a fixed type, eg, the program state.

\( \alpha S = \text{State} \rightarrow \alpha \ast \text{State} \)

\( \text{unit}_S \ a \ s = (a, s) \)

\( (g \circ_s f) \ a \ s = \text{let} \ (b, s') = f \ a \ s \ \text{in} \ g \ b \ s' \)

Proof tedious, but

\( \text{curry} \ g \ x \ y = g \ (x, y) \quad \text{unc} \ f \ (x, y) = f \ x \ y \)

(mutually inverse, and so 1-1)

for (A1) and (A2):

\( \text{unc} \ (f \circ_s \text{unit}_S) = \text{unc} \ f \circ \text{unc} \text{unit}_S = \text{unc} \ f \circ \text{id} = \text{unc} \ f \)

\( \text{unc} \ (\text{unit}_S \circ_s f) = \text{unc} \text{unit}_S \circ \text{unc} \ f = \text{id} \circ \text{unc} \ f = \text{unc} \ f \)

for (A3): \( \text{unc} \ (h \circ_s (g \circ_s f)) = \text{unc} \ h \circ (\text{unc} \ g \circ \text{unc} \ f) = \)

\( (\text{unc} \ h \circ \text{unc} \ g) \circ \text{unc} \ f = \text{unc} \ ((h \circ_s g) \circ_s f) \)
The Compound State Monad

Let $M$ be any monad. Define $\alpha S_M = \text{State} \to (\alpha \ast \text{State}) M$.

We can define $\circ_{SM}$ and $\text{unit}_{SM}$ by

$$\text{unc } (g \circ_{SM} f) = \text{unc } g \circ_M \text{unc } f$$

$$\text{unc } \text{unit}_{SM} = \text{unit}_{M}$$

Then the proofs are easy, using corresponding rules for monad $M$.

for (A1) and (A2):

$$\text{unc } (f \circ_{SM} \text{unit}_{SM}) = \text{unc } f \circ_M \text{unc } \text{unit}_{SM} = \text{unc } f \circ_M \text{unit}_{M} = \text{unc } f$$

$$\text{unc } (\text{unit}_{SM} \circ_{SM} f) = \text{unc } \text{unit}_{SM} \circ_M \text{unc } f = \text{unit}_{M} \circ_M \text{unc } f = \text{unc } f$$

for (A3):

$$\text{unc } (h \circ_{SM} (g \circ_{SM} f)) = \text{unc } h \circ_M (\text{unc } g \circ_M \text{unc } f) =$$

$$\text{unc } (h \circ_M \text{unc } g) \circ_M \text{unc } f = \text{unc } ((h \circ_{SM} g) \circ_{SM} f)$$

Other definitions more complicated, and proofs correspondingly so.
A free theorem ??

Still need to prove (A4),

\[(h \circ \text{id}) \circ f = h \circ f \quad f : \alpha \rightarrow \beta M \quad h : \beta \rightarrow \gamma M\]

Can we use Wadler’s “free theorems”?

Idea is that \(h \circ - : (\alpha \rightarrow \beta M) \rightarrow \alpha \rightarrow \gamma M\) is polymorphic in \(\alpha\), so in applying \(h \circ f\) to \(a : \alpha\), can only apply \(f\) to \(a\), and then do something else, call it \(g\).

That is, \(h \circ f = g \circ f\), so

\[(h \circ \text{id}) \circ f = (g \circ \text{id}) \circ f = g \circ f = h \circ f\]

Is this valid?
Other Monads

List monad: $\alpha L = \alpha \text{ list}$

Reader monad: $\alpha R = \text{param} \to \alpha$

Writer monad: $\alpha W = \alpha \times \text{output}$

Error monad: $\alpha E = \alpha \text{ option}$  (SOME $a$ for success, NONE for failure)

These can form compound monads with an arbitrary monad $M$ in different ways: $\alpha Mr, \alpha Wm, \alpha Em$

$\alpha E$ can represent termination of a program, in a final state, or non-termination

$\alpha E L$ represents corresponding non-deterministic program operation

$\alpha L E$ also represents such a program for considering total correctness:
if it may fail to terminate then never mind what else it might do, it is a failure.

$\alpha L E$ is also a compound monad
Compound Monads via Partial Extension

compound monad type is \((\alpha N)M = \alpha NM\). To define a compound monad \(NM\), need \(\text{ext}_{NM}\), “extending” a function \(f\) from a “smaller” domain, \(\alpha\), to a “larger” one, \(\alpha NM\).

Consider a “partial extension” function \(\text{pext}\) which does part of this job:

\[
\begin{align*}
\text{ext}_{NM} &: (\alpha \to \beta NM) \to (\alpha NM \to \beta NM) \\
\text{pext} &: (\alpha \to \beta NM) \to (\alpha N \to \beta NM)
\end{align*}
\]

Then \(\text{ext}_{NM} f = \text{ext}_M (\text{pext} f)\).

Rules (E1K) to (E3K) are enough to define \(NM\).

About \(\text{comp}_{NM}\) or \(\text{unit}_{NM}\), assume only they have the right types.

Need not assume that \(N\) is a monad.
Monad rules for a compound monad using $pext$

\[
\begin{align*}
\text{pext } f \; &\odot_M \; \text{unit}_{NM} = f & \text{(E1K)} \\
\text{pext } \text{unit}_{NM} & = \text{unit}_M & \text{(E2K)} \\
\text{pext } (g \; \odot_{NM} \; f) & = \text{pext } g \; \odot_M \; \text{pext } f & \text{(E3K)}
\end{align*}
\]

\[
\begin{align*}
\text{kjoin} & = \text{pext } \text{unit}_M & \text{(E4K)} \\
\text{kmap } f & = \text{pext } (\text{unit}_{NM} \; \odot_M \; f) & \text{(E5K)}
\end{align*}
\]

\[
\begin{align*}
g \; \odot_{NM} \; f & = \text{pext } g \; \odot_M \; f & \text{(E6K)}
\end{align*}
\]

These are just the rules needed for a monad $N$ in $\mathcal{K}_M$
Correspondence between monad $N$ and monad $N$ in $\mathcal{K}_M$

\[ \begin{align*}
  id : \alpha & \rightarrow \alpha \\
  unit_N : \alpha & \rightarrow \alpha N \\
  map_N : (\alpha \rightarrow \beta) & \rightarrow \alpha N \rightarrow \beta N \\
  join_N : \alpha NN & \rightarrow \alpha N \\
  ext_N : (\alpha \rightarrow \beta N) & \rightarrow \alpha N \rightarrow \beta N \\
  unit_M : \alpha & \rightarrow \alpha M \\
  unit_{NM} : \alpha & \rightarrow \alpha NM \\
  kmap : (\alpha \rightarrow \beta M) & \rightarrow \alpha N \rightarrow \beta NM \\
  kjoin : \alpha NN & \rightarrow \alpha NM \\
  pext : (\alpha \rightarrow \beta NM) & \rightarrow \alpha N \rightarrow \beta NM \\
  \text{ext}_N g & = g \odot_N id \\
  g \odot_N f & = \text{ext}_N g \circ f \\
  join_N & = \text{ext}_N id \\
  map_N f & = \text{ext}_N (unit_N \circ f) \\
  \text{ext}_N f & = join_N \circ \text{map}_N f \\
  h \odot_N f & = (h \odot_N id) \circ f \\
  \text{pext } g & = g \odot_{NM} unit_M \\
  g \odot_{NM} f & = \text{pext } g \odot_M f \\
  kjoin & = \text{pext } unit_M \\
  kmap f & = \text{pext } (unit_{NM} \odot_M f) \\
  \text{pext } f & = kjoin \odot_M kmap f \\
  h \odot_{NM} f & = (h \odot_{NM} unit_M) \odot_M f
\end{align*} \]
To show $NM$ is a monad, using the pext rules

\[
\begin{align*}
  f \circ_{NM} \text{unit}_{NM} &= f \\
  \text{unit}_{NM} \circ_{NM} f &= f \\
  h \circ_{NM} (g \circ_{NM} f) &= (h \circ_{NM} g) \circ_{NM} f \\
  (h \circ_{NM} \text{unit}_M) \circ_M f &= h \circ_{NM} f
\end{align*}
\]

To show $NM$ is a monad, want namely $(A1NM)$ to $(A4NM)$.

But $(A1NM)$ to $(A3NM)$ same as $(A1K)$ to $(A3K)$; only $(A4NM)$ is different.

So need only $(A4NM)$; to get it, have both $(A4K)$ and $(A4M)$.

\[
\begin{align*}
  (h \circ_{NM} \text{id}) \circ f &= h \circ_{NM} f \\
  (h \circ_M \text{id}) \circ f &= h \circ_M f
\end{align*}
\]

\[
\begin{align*}
  (h \circ_{NM} \text{id}) \circ f &= ((h \circ_{NM} \text{unit}_M) \circ_M \text{id}) \circ f \\
  &= (h \circ_{NM} \text{unit}_M) \circ_M f = h \circ_{NM} f
\end{align*}
\]
What characterises such compound monads?

Such compound monads $NM$ satisfy

\[
\begin{align*}
\text{ext}_{NM} f &= \text{ext}_M (pext f) \quad \text{(EC)} \\
pext f &= \text{ext}_{NM} f \circ \text{unit}_M \quad \text{(PE)} \\
\text{ext}_M (\text{ext}_{NM} f) &= \text{ext}_{NM} f \circ \text{join}_M \quad \text{(J1S)}
\end{align*}
\]

Note, (J1S) of the form of Theorem 1(i).

Conversely, if $M$ and $NM$ are monads, and (J1S) holds, then $\odot_{NM}$ also defines a monad in $K_M$, and, using (PE) to define $pext$, (EC) holds.

Proof uses that (A1K) to (A3K) same as (A1NM) to (A3NM); shows (A4K) and (EC) from (J1S) using Theorem 1.
A more general set of rules

Three more functions of the following types:

\[ dunit : \alpha M \to \alpha NM \]
\[ dmap : (\alpha \to \beta M) \to (\alpha NM \to \beta NM) \]
\[ djoin : \alpha NNM \to \alpha NM \]

\[ dmap \text{ unit}_M = \text{id} \] \hspace{1cm} (G1)
\[ dmap (f \circ h) = dmap f \circ \text{map}_NM h \] \hspace{1cm} (G2)
\[ dmap f \circ \text{unit}_NM = dunit \circ f \] \hspace{1cm} (G3)
\[ djoin \circ dmap (dmap f) = dmap f \circ \text{join}_NM \] \hspace{1cm} (G4)
\[ djoin \circ \text{unit}_NM = \text{id} \] \hspace{1cm} (G5)
\[ djoin \circ dmap \text{ unit}_NM = \text{id} \] \hspace{1cm} (G6)
\[ djoin \circ dmap djoin = djoin \circ \text{join}_NM \] \hspace{1cm} (G7)
\[ \text{ext}_NM f = djoin \circ dmap f \] \hspace{1cm} (G8)
These also give a monad \( NM \)

**Theorem 3** Assume rules (G1) to (G8). Then \( \text{ext}_{NM}, \text{join}_{NM}, \text{map}_{NM} \) and \( \text{unit}_{NM} \) give a monad \( NM \), where also

\[
\begin{align*}
  \text{djoin} &= \text{ext}_{NM} \text{unit}_M \\
  \text{dmap} f &= \text{ext}_{NM} (\text{dunit} \circ f) \\
  \text{unit}_{NM} &= \text{dunit} \circ \text{unit}_M \\
  \text{map}_{NM} f &= \text{dmap} (\text{unit}_M \circ f)
\end{align*}
\]

(G9) \hspace{2cm} \text{(G10)} \hspace{2cm} \text{(G11)} \hspace{2cm} \text{(G12)}

Conversely, for a compound monad \( NM \), when is this construction applicable?

**Theorem 4** Assume that \( NM \) is a monad. Also assume that rules (G5) and (G9) to (G11) hold. Then the remaining rules among (G1) to (G8) hold.
When is the construction applicable?

How to use Theorem 4? Assume (UC).

\[
unit_{NM} f = unit_M (unit_N f) \quad \text{(UC)}
\]

\[
dunit = map_M unit_N \quad \text{(DU)}
\]

\[
ext_{NM} unit_M \circ map_M unit_N = id \quad \text{(G5')}
\]

If (J1S) and the \textit{pext} construction hold, then define functions \textit{dunit}, \textit{dmap} and \textit{djoin} by (DU), (G10) and (G9).

Then (G11) holds by (3M), and (G5) becomes (G5') which holds: the proof uses \[
ext_{NM} f = ext_M (pext f).
\]

So Theorem 4 applies.
On the other hand...

\[ \text{ext}_{NM} (\text{map}_M \text{join}_N) = \text{map}_M \text{join}_N \circ \text{join}_{NM} \]  \hfill (J2')

Note (J2') is also of the form of Theorem 1(i).

If \( N \) is a monad and \( M \) a premonad, and (J2') holds, then
\[ \text{ext}_{NM} \text{unit}_M = \text{map}_M \text{join}_N \] (proved from Theorem 1).

In this case (G5)/(G5') also hold, by a seemingly different proof.

\[ \text{ext}_{NM} \text{unit}_M \circ \text{map}_M \text{unit}_N = \text{map}_M \text{join}_N \circ \text{map}_M \text{unit}_N \]
\[ = \text{map}_M (\text{join}_N \circ \text{unit}_N) = \text{map}_M \text{id} = \text{id} \]

So again Theorem 4 applies.

Common feature: \( \text{ext}_{NM} \text{unit}_M \) is of the form \( \text{ext}_M f \), so Theorem 1 applies.
When both (J1S) and (J2') hold

If both (J1S) and (J2') hold, and both $M$ and $N$ are monads, then we have a distributive law for the monads $M$, $N$ and $NM$. 