Monad algebras
Monad algebras

- An *algebra* of a monad \((T, \eta, \mu)\) is an object \(X\) with a map \(\xi : TX \to X\) such that

\[
\begin{align*}
X & \xrightarrow{\eta_X} TX \\
& \searrow \downarrow \nearrow \\
TX & \xrightarrow{\xi} X
\end{align*}
\]

\[
\begin{align*}
T(TX) & \xrightarrow{T\xi} TX \\
& \searrow \downarrow \nearrow \\
TX & \xrightarrow{\xi} X
\end{align*}
\]

- A *map* between two algebras \((Y, \chi)\) and \((X, \xi)\) is a map \(h\) such that

\[
\begin{align*}
TY & \xrightarrow{T h} TX \\
& \searrow \downarrow \nearrow \\
Y & \xrightarrow{h} X
\end{align*}
\]

- The algebras of the monad and maps between them form a category \(\text{Alg}(T)\), called the *Eilenberg-Moore category*, with an obvious forgetful functor \(U : \text{Alg}(T) \to C\).
Kleisli triple algebras

- A variation of algebras fitting more smoothly with Kleisli triples is this.

- A *algebra* of a Kleisli triple \((T, \eta, (-)^*)\) (*a Mendler-style algebra, an algebra in extension form, no-iteration form*) is given by
  - an object \(X\),
  - a family of maps \((-)^+_Y : \mathcal{C}(Y, X) \to \mathcal{C}(TY, X)\) indexed by \(Y \in |\mathcal{C}|\)
  such that
    - if \(f : Y \to X\), then \(f^+ \circ \eta_Y = f\)
    - if \(k : Z \to TY, f : Y \to X\), then \((f^+ \circ k)^+ = f^+ \circ k^* : TZ \to X\)

- Naturality of \((-)^+\) is not required, it follows.

- There's also the correct concept of Kleisli triple algebra map. (Definition omitted.)
Monad algebras = Kleisli triple algebras

- Algebras of monads/Kleisli triples with the same carrier $X$ are in a bijection.

- This is again crucially by the Yoneda lemma.

$$\begin{align*}
  TX \to X \\
  C(Y, X) \to C(TY, X) \text{ nat. in } Y
\end{align*}$$

- From $\xi$, one defines $(-)^+$ by $f^+ = \xi \circ Tf$.
- From $(-)^+$, one defines $\xi$ by $\xi = \text{id}_X^+$.

- The respective categories are isomorphic.
FP intuition

- An algebra of a monad $T$ with carrier $X$ is a “handler” of computations of values of the type $X$ (and only of that type!).

- $\xi : TX \to X$ –
  a value of $X$ can be extracted from a computation of values of $X$

- $(-)^+_{Ty} : C(Y, X) \to C(TY, X)$ –
  given a way $f : Y \to X$ to “observe” values of $Y$ as values of $X$, $f^+ : TY \to X$ is a way of observing computations of values of $Y$
Eilenberg-Moore adjunction

- In the opposite direction of $U : \text{Alg}(T) \to C$ there is a functor $L : C \to \text{Alg}(T)$ defined by
  - $LX = (TX, \mu_X)$,
  - $Lf = Tf : (TY, \mu_Y) \to (TX, \mu_X)$ for $f : Y \to X$.
- $L$ is left adjoint to $U$.
- $U \cdot L = T$. Indeed,
  - $U(LX) = U(TX, \mu_X) = TX$,
  - if $f : Y \to X$, then $U(Lf) = U(Tf) = Tf$.
- The unit of the adjunction is $\eta$.
- The E-M resolution of a monad is its final resolution.
Algebras of exceptions monads

- Algebras of the exceptions monad $TX = E + X$ are (by definition) objects $X$ with a map $\xi : E + X \rightarrow X$ subject to 2 equations.

- They are in a bijection with pairs of an object $X$ and map $E \rightarrow X$.

- The E-M category of this monad is isomorphic to the coslice category $E/C$.

  [FP intuition] These are handlers for exceptional computations!

- To able to extract a value from any given exceptional computation, you must know how to deal with the exception case.
Algebras of reader monads

- Algebras of the reader monad \( TX = S \Rightarrow X \) are (by definition) objects \( X \) with a map \( \text{get} : S \Rightarrow X \rightarrow X \) such that
  - \( \text{get} (\lambda s. x) = x \)
  - \( \text{get} (\lambda s. \text{get} (\lambda s'. f s s')) = \text{get} (\lambda s. f s s) \)
Algebras of state monads

- The E-M category of the state monad $TX = S \Rightarrow S \times X$ is isomorphic to the category of mnemoids.

- An algebra of this monad is an object $X$ with a map $\text{getput} : S \Rightarrow S \times X \rightarrow X$ such that
  \[
  x = \text{getput} (\lambda s. (s, x))
  \]
  \[
  \text{getput} (\lambda s. \text{let } (s', g) = f s \text{ in } (s', \text{getput } g)) =
  \]
  \[
  \text{getput} (\lambda s. \text{let } (s', g) = f s \text{ in } g s')
  \]

- A mnemoid is an object $X$ with maps $\text{get} : S \Rightarrow X \rightarrow X$ and $\text{put} : S \times X \rightarrow X$ such that
  \[
  x = \text{get} (\lambda s. \text{put } (s, x))
  \]
  \[
  \text{put} (s, \text{get } f) = \text{put} (s, f s)
  \]
  \[
  \text{put} (s, \text{put } (s', x)) = \text{put } (s', x)
  \]

- From $\xi$, one constructs $\text{get}$, $\text{put}$ by $\text{get } f = \xi (\lambda s. (s, f s))$, $\text{put } (s, x) = \xi (\lambda_. (s, x))$.

- From $\text{get}$, $\text{put}$, one obtains $\xi$ by $\xi f = \text{get } (\lambda s. \text{put } (f s))$. 
Algebras of list monads

- The E-M category of the standard list monad is isomorphic to that of monoids, i.e., objects $X$ with maps $1 \to X$ and $X \times X \to X$ satisfying left and right unitality and associativity.

- It is therefore also called the *free monoids monad*.

- The E-M category of the alternative list monad is in a bijection with semigroups with zero.

- A *semigroup with zero* is an object $X$ with maps $1 \to X$ and $X \times X \to X$ satisfying left and right zeroness and associativity.
Algebras of free functor-algebras monads

- The E-M category $\text{Alg}(F^*)$ of the monad $F^*$ of free algebras of a functor $F$ is isomorphic to the category $\text{alg}(F)$ of algebras of $F$

\[
\begin{array}{ccc}
\text{Alg}(F^*) & \cong & \text{alg}(F) \\
& \searrow & \\
& C & \nearrow \\
& U & \\
\end{array}
\]

- For $FX = X \times X$, algebras with carrier $X$ of the monad $F^*$ are maps $\mu Z. X + Z \times Z \to X$ subject to two equations.

- They are in bijection with algebras with carrier $X$ of the functor $F$, which are maps $X \times X \to X$ subject to no conditions (magnas).

- A monad with this property is said to be algebraically free on $F$. 
Monad maps
Monad maps

- A monad map between monads $T$, $T'$ on a category $C$ is a natural transformation $\tau : T \rightarrow T'$ satisfying

\[
\begin{align*}
X & \xrightarrow{\eta_X} TX & & T(TX) & \xrightarrow{T\tau_X} T'(TX) & \xrightarrow{T'\tau_X} T'(T'X) \\
& \downarrow{\eta_X} & & \downarrow{\eta'_X} & & \downarrow{\mu_X} & & \downarrow{\mu'_X} \\
TX & \xrightarrow{\tau_X} T'X & & TX & \xrightarrow{\tau_X} & T'X
\end{align*}
\]

- Monads on $C$ and maps between them form a category $\text{Monad}(C)$.

- $\text{Monad}(C)$ is the category of monoids in the (strict) monoidal category $([C, C], \text{Id}_C, \cdot)$.
Kleisli triple maps

- A map between two Kleisli triples $T$, $T'$ is a family of maps $\tau_X : TX \to T'X$ indexed by $X \in |C|$ such that
  - $\tau_X \circ \eta_X = \eta'_X$,
  - if $k : X \to TY$, then $\tau_Y \circ k^* = (\tau_Y \circ k)^* \circ \tau_X$.

- Naturality of $\tau$ is not required, but it follows.

- Kleisli triples on $C$ and maps between them form a category isomorphic to $\mathbf{Monad}(C)$. 
Maps between exceptions, reader, writer monads

- Monad maps between the exception monads for sets $E$, $E'$ are in a bijection with pairs of maps $1 \to E' + 1$ and $E \to E'$.

- Monad maps between the reader monads for sets $S$, $S'$ are in a bijection with maps between $S'$, $S$.

- Monad maps between the writer monads for monoids $(P, o, \oplus)$ and $(P', o', \oplus')$ are in a bijection with homomorphisms between these monoids.
Maps between state monads

- The monad maps between the state monads for $S$ and $S_0$ are in a bijection with *(very well-behaved) lenses*.

- These are pairs of maps $\text{coget} : S_0 \rightarrow S$, $\text{coput} : S_0 \times S \rightarrow S_0$ such that
  - $s_0 = \text{coput} (s_0, \text{coget} s_0))$,
  - $\text{coget} (\text{coput} (s_0, s)) = s$,
  - $\text{coput} (\text{coput} (s_0, s), s') = \text{coput} (s_0, s')$. 
Free functor-algebras monads are free

- The monad $F^*$ of free algebras of a functor $F$ (the algebraically-free monad on $F$), if it exists, is the free monad on $F$.

\[
\begin{array}{c}
\text{Monad}(C) \\
(\cdot)^* \quad \circlearrowleft \quad \downarrow u \\
\downarrow \downarrow \\
[C, C] \\
\end{array}
\]

\[
F^* \to R \\
\frac{F \to UR}{F \to U R}
\]

- (Use the full subcategory of $[C, C]$ of those functors for which $(-)^*$ exists.)

- If a monad $T$ is free on $F$, it need not be algebraically-free on $F$.
- A monad $T$ is free on $F$ iff $T \cong \mu H. \text{Id} + F \cdot H$.
- It is algebraically free iff $TX \cong \mu Z. X + F(TX)$. This is generally a stronger condition.
Maps to continuation monads

- Let \( \text{xCnt}^R \) be the external continuation monad for \( R \)
  \( (\text{xCnt}^R X = C(X, R) \triangleright R) \).
- Monad maps between an arbitrary monad \( T \) and the monad \( \text{xCnt}^R \)
  are in a bijection with algebras of \( T \) with carrier \( R \).
- Yoneda strikes again. :-)

\[
\begin{align*}
  TR & \to R \\
  C(X, R) & \to C(TX, R) \text{ nat. in } X \\
  TX & \to C(X, R) \triangleright R \text{ nat. in } X
\end{align*}
\]

- Let \( Cnt^R \) be the continuation monad for \( R \), which is strong.
- Strong monad maps between an arbitrary strong monad \( T \) and \( Cnt^R \)
  are in a bijection with algebras \( T \) with carrier \( R \).
Monad maps vs. functors between Kleisli categories

There is a bijection between monad maps $\tau : T \to T'$ and functors $V : \text{Kl}(T) \to \text{Kl}(T')$ such that

$$\text{Kl}(T) \xrightarrow{V} \text{Kl}(T')$$

This is defined by

- $V X = X$,
- $V k = Y \xrightarrow{k} TX \xrightarrow{\tau_X} T'X$ for $k : Y \to TX$

and

- $\tau_X = V(TX \xrightarrow{\text{id}_TX} TX) : TX \to T'X$. 

Monad maps vs. functors between $E$-$M$ categories

- There is a bijection between monad maps $\tau : T \to T'$ and functors $V : \text{Alg}(T') \to \text{Alg}(T)$ such that

$$
\begin{array}{c}
\text{Alg}(T') \\
\downarrow V \\
\text{Alg}(T)
\end{array}
\begin{array}{c}
\downarrow U' \\
\downarrow U \\
\downarrow C
\end{array}
$$

(Note the reversed direction.)

- This is defined by
  - $V(X, \xi) = (X, TX \xrightarrow{\tau_X} T'X \xrightarrow{\xi} X)$,
  - $Vh = h : (Y, \chi \circ \tau_Y) \to (X, \xi \circ \tau_X)$ for $h : (Y, \chi) \to (X, \xi)$

and
  - $\tau_X = \text{let } (T'X, \zeta) = V(T'X, \mu'_X) \text{ in } \zeta \circ T\eta'_X$. 