Monads and More: Part 1

Tarmo Uustalu, Institute of Cybernetics, Tallinn

University of Nottingham, 14–18 May 2007
University of Udine, 2–6 July 2007
Outline

- Monads and why they matter for a working functional programmer: monads, Kleisli categories, monadic computation, strong and commutative monads, monadic semantics
- Combining monads: monads from adjunctions, distributive laws, the coproduct of monads
- Finer and coarser: Lawvere theories, arrows and Freyd categories
- Comonadic notions of computation: comonads and coKleisli categories, comonadic computation, in particular dataflow computation, lax/strong symmetric monoidal comonads, comonadic semantics
- Notions of computation on trees
Prerequisites

- Basics of functional programming and typed lambda calculi

- From category theory:
  - functors, natural transformations
  - adjunctions
  - symmetric monoidal (closed) categories
  - Cartesian (closed) categories, coproducts
  - initial algebra, final coalgebra of a functor
Monads

- A monad on a category $C$ is given by a
  - a functor $T : C \to C$ (the underlying functor),
  - a natural transformation $\eta : \text{Id}_C \to T$ (the unit),
  - a natural transformation $\mu : TT \to T$ (the multiplication)

satisfying these conditions:

$$
\begin{align*}
TA & \xrightarrow{\eta_TA} TTA \\
TTA & \xrightarrow{\mu_A} TA \\
TTTA & \xrightarrow{\mu_TA} TTA \\
TTTA & \xrightarrow{\mu_A} TA
\end{align*}
$$

- This definition says that $(T, \eta, \mu)$ is a monoid in the endofunctor category $[C, C]$. 
An alternative formulation: Kleisli triples

- A more combinatory formulation is the following.
- A *monad* (*Kleisli triple*) is given by
  - an object mapping $T : \mathcal{C} \rightarrow \mathcal{C}$,
  - for any object $A$, a map $\eta_A : A \rightarrow TA$,
  - for any map $k : A \rightarrow TB$, a map $k^* : TA \rightarrow TB$ (the *Kleisli extension* operation)

satisfying these conditions:
- if $k : A \rightarrow TB$, then $k^* \circ \eta_A = k$,
- $\eta_A^* = \text{id}_{TA}$,
- if $k : A \rightarrow TB$, $\ell : B \rightarrow TC$, then $(\ell^* \circ k)^* = \ell^* \circ k^*$.

(Notice there are no explicit functoriality and naturality conditions.)
Monads vs. Kleisli triples

- There is a bijection between monads and Kleisli triples.
- Given $T$, $\eta$, $\mu$, one defines
  - if $k : A \to TB$, then $k^* = \text{df} \quad TA \xrightarrow{Tk} TTB \xrightarrow{\mu_B} TB$.
  - Given $T$ (on objects only), $\eta$ and $-^*$, one defines
    - if $f : A \to B$, then
      $$Tf = \text{df} \quad (A \xrightarrow{f} B \xrightarrow{\eta_B} TB)^* : TA \to TB,$$
    - $\mu_A = \text{df} \quad (TA \xrightarrow{id_{TA}} TA)^* : TTA \to TA.$
Kleisli category of a monad

- A monad \( T \) on a category \( C \) induces a category \( \text{Kl}(T) \) called the *Kleisli category* of \( T \) defined by
  - an object is an object of \( C \),
  - a map of from \( A \) to \( B \) is a map of \( C \) from \( A \) to \( TB \),
  - \( \text{id}^T_A = \text{df} A \xrightarrow{\eta_A} TA \),
  - if \( k : A \to^T B, \ell : B \to^T C \), then
    \( \ell \circ^T k = \text{df} A \xrightarrow{k} TB \xrightarrow{T\ell} TTC \xrightarrow{\mu_C} TC \)
- From \( C \) there is an identity-on-objects *inclusion functor* \( J \) to \( \text{Kl}(T) \), defined on maps by
  - if \( f : A \to B \), then
    \( Jf = \text{df} A \xrightarrow{f} B \xrightarrow{\eta_B} TB = A \xrightarrow{\eta_A} TA \xrightarrow{Tf} TB \).
Computational interpretation

- Think of $C$ as the category of pure functions and of $TA$ as the type of effectful computations of values of a type $A$.
- $\mathbf{KI}(T)$ is then the category of effectful functions.
- $\eta_A : A \to TA$ is the identity function on $A$ viewed as trivially effectful.
- $Jf : A \to TB$ is a general pure function $f : A \to B$ viewed as trivially effectful.
- $\mu_A : TTA \to TA$ flattens an effectful computation of an effectful computation.
- $k^* : TA \to TB$ is an effectful function $k : A \to TB$ extended into one that can input an effectful computation.
Examples

- **Exceptions monad:**
  - \( TA =_{df} A + E \) where \( E \) is some object (of exceptions),
  - \( \eta_A =_{df} A \xrightarrow{\text{inl}} A + E, \)
  - \( \mu_A =_{df} (A + E) + E \xrightarrow{[\text{id,inr}]} A + E, \)
  - if \( k : A \rightarrow B + E \), then \( k^* =_{df} A + E \xrightarrow{[k,\text{inr}]} B + E. \)

- **Output monad:**
  - \( TA =_{df} A \times E \) where \((E, e, m)\) is some monoid (of output traces), e.g., the type of lists of a fixed element type with nil and append,
  - \( \eta_A =_{df} A \xrightarrow{\text{ur}} A \times 1 \xrightarrow{\text{id} \times e} A \times E, \)
  - \( \mu_A =_{df} (A \times E) \times E \xrightarrow{a} A \times (E \times E) \xrightarrow{\text{id} \times m} A \times E, \)
  - if \( k : A \rightarrow B \times E \), then
    \[ k^* =_{df} A \times E \xrightarrow{k \times \text{id}} (B \times E) \times E \xrightarrow{a} B \times (E \times E) \xrightarrow{\text{id} \times m} B \times E. \]
Reader monad:

- $TA =_{df} E \Rightarrow A$ where $E$ is some object (of environments),
- $\eta_A =_{df} \Lambda(A \times E \xrightarrow{\text{fst}} A)$,
- $\mu_A =_{df} \Lambda((E \Rightarrow (E \Rightarrow A)) \times E \xrightarrow{(ev, snd)} (E \Rightarrow A) \times E \xrightarrow{ev} A)$,
- if $k : A \rightarrow E \Rightarrow B$, then $k^* =_{df} \Lambda((E \Rightarrow A) \times E \xrightarrow{(ev, snd)} A \times E \xrightarrow{k \times id} (E \Rightarrow B) \times E \xrightarrow{ev} B)$. 
Side-effect monad:

- \( TA =_{df} S \Rightarrow A \times S \) where \( S \) is some object (of states),
- \( \eta_A =_{df} \Lambda(A \times S \xrightarrow{id} A \times S) \),
- \( \mu_A =_{df} \Lambda(S \Rightarrow ((S \Rightarrow A \times S) \times S) \times S \xrightarrow{ev} (S \Rightarrow A \times S) \times S \xrightarrow{ev} A \times S) \),
- if \( k : A \rightarrow S \Rightarrow B \times S \), then \( k^* =_{df} \Lambda(((S \Rightarrow A \times S) \times S \xrightarrow{ev} A \times S) \xrightarrow{k \times id} (S \Rightarrow B \times S) \times S \xrightarrow{ev} B \times S) \).

Continuations monad:

- \( TA =_{df} (A \Rightarrow R) \Rightarrow R \) where \( R \) is some object (of answers),
- \( \eta_A =_{df} \Lambda(A \times (A \Rightarrow R) \xrightarrow{c} (A \Rightarrow R) \times R \xrightarrow{ev} R) \),
- if \( k : A \rightarrow (B \Rightarrow R) \Rightarrow R \), then
  \( k^* =_{df} \Lambda(((A \Rightarrow R) \Rightarrow R) \times (B \Rightarrow R) \xrightarrow{id \times \Lambda(\Lambda^{-1}(k) \circ c)} ((A \Rightarrow R) \Rightarrow R) \times (A \Rightarrow R) \xrightarrow{ev} R) \).
Strong functors

- A strong functor on a category \((C, I, \otimes)\) is given by
  - an endofunctor \(F\) on \(C\),
  - together with a natural transformation
    \(\text{sl}_{A,B} : A \otimes FB \to F(A \otimes B)\) (the (tensorial) strength)

satisfying

\[
\begin{array}{c}
I \otimes FA \xrightarrow{\text{sl}_{I,A}} F(I \otimes A) \\
\downarrow \text{id}_{FA} \quad \quad \quad \downarrow \text{Ful}_{A} \\
FA \quad \quad \quad FA
\end{array}
\]

\[
\begin{array}{c}
(A \otimes B) \otimes FC \xrightarrow{\text{sl}_{A\otimes B,C}} F((A \otimes B) \otimes C) \\
\downarrow a_{A,B,FC} \quad \quad \quad \downarrow F_{a_{A,B,C}} \\
A \otimes (B \otimes FC) \xrightarrow{\text{id}_A \otimes \text{sl}_{B,C}} A \otimes F(B \otimes C) \xrightarrow{\text{sl}_{A,B\otimes C}} F(A \otimes (B \otimes C))
\end{array}
\]
A strong natural transformation between two strong functors \((F, \text{sl}), (G, \text{sl}')\) is a natural transformation \(\tau : F \to G\) satisfying

\[
\begin{align*}
A \otimes FB & \xrightarrow{\text{sl}_{A,B}} F(A \otimes B) \\
\text{id}_{A \otimes T_B} & \downarrow \quad \downarrow \quad \tau_{A \otimes B} \\
A \otimes GB & \xrightarrow{\text{sl}'_{A,B}} G(A \otimes B)
\end{align*}
\]
A strong monad on a monoidal category \((\mathcal{C}, I, \otimes)\) is a monad \((T, \eta, \mu)\) together with a strength \(sl\) for \(T\) for which \(\eta\) and \(\mu\) are strong, i.e., satisfy

\[
\begin{align*}
A \otimes B & \cong A \otimes B \\
\text{id}_{A \otimes B} & \downarrow \\
A \otimes TB & \rightarrow T(A \otimes B) \\
\eta_{A \otimes B} & \\
A \otimes TTB & \xrightarrow{sl_{A,B}^{A \otimes B}} T(A \otimes TB) \\
T \times T & \rightarrow TT(A \otimes B) \\
\mu_{A \otimes B} & \\
A \otimes TB & \rightarrow T(A \otimes B) \\
\end{align*}
\]

(Note that \(\text{Id}\) is always strong and, if \(F, G\) are strong, then \(GF\) is strong.)
Commutative monads

- If \((C, I, \otimes)\) is symmetric monoidal, then a strong functor \((F, sl)\) is actually bistrong: it has a *costrength* \(sr_{A,B} : FA \otimes B \rightarrow F(A \otimes B)\) with properties symmetric to those of a strength defined by

  \[ sr_{A,B} = \text{df} \quad FA \otimes B \xrightarrow{c_{FA,B}} B \otimes FA \xrightarrow{sl_{B,A}} F(B \otimes A) \xrightarrow{F_{CB,A}} F(A \otimes B) \]

- A bistrong monad \((T, sl, sr)\) is called *commutative*, if it satisfies

\[
\begin{align*}
  TA \otimes TB & \xrightarrow{sl_{TA,B}} T(TA \otimes B) \xrightarrow{Tsr_{A,B}} TT(A \otimes B) \\
  T(A \otimes TB) & \xrightarrow{Tsl_{A,B}} TT(A \otimes B) \\
  TT(A \otimes B) & \xrightarrow{\mu_{A \otimes B}} T(A \otimes B)
\end{align*}
\]
Examples

- **Exceptions monad:**
  \[ TA =_{df} A + E \text{ where } E \text{ is an object,} \]
  \[ sl_{A,B} =_{df} A \times (B + E) \xrightarrow{\text{dr}} A \times B + A \times E \xrightarrow{\text{id+snd}} A \times B + E. \]

- **Output monad:**
  \[ TA =_{df} A \times E \text{ where } (E, e, m) \text{ is a monoid,} \]
  \[ sl_{A,B} =_{df} A \times (B \times E) \xrightarrow{a^{-1}} (A \times B) \times E. \]

- **Reader monad:**
  \[ TA =_{df} E \Rightarrow A \text{ where } E \text{ is an object,} \]
  \[ sl_{A,B} =_{df} \Lambda((A \times (E \Rightarrow B)) \times E \xrightarrow{a} A \times ((E \Rightarrow B) \times E) \xrightarrow{\text{id} \times \text{ev}} A \times B). \]
Tensorial vs. functorial strength

- A *functorially strong functor* on a monoidal closed category \((\mathcal{C}, I, \otimes, \to)\) is an endofunctor \(F\) on \(\mathcal{C}\) with a natural transformation \(fs_{A,B} : A \to B \to FA \to FB\) internalizing the functorial action of \(F\).

- There is a bijective correspondence between tensorially and functorially strong endofunctors, in fact an equivalence between their categories.

- Given \(fs\), one defines \(sl\) by

\[
sl_{A,B} =_{df} A \otimes FB \xrightarrow{\text{coev} \otimes \text{id}} (B \to A \otimes B) \otimes FB \xrightarrow{\Lambda^{-1}(fs)} F(A \otimes B)
\]

- Given \(sl\), one defines \(fs\) by

\[
fs_{A,B} =_{df} \Lambda((A \to B) \otimes FA \xrightarrow{sl} F((A \to B) \otimes A) \xrightarrow{Fev} FB)
\]
On $\text{Set}$, every monad is $(1, \times)$ strong

- Any endofunctor on $\text{Set}$ has a unique functorial strength and any natural transformation between endofuctors on $\text{Set}$ is functorially strong.
- Hence any monad on $\text{Set}$ is both functorially and tensorially strong.
Effects

- Of course we want the Kleisli category of a monad to contain more maps than the base category.
- To describe those, we must single out some proper sources of effectfulness. How to choose those is a topic on its own.
- E.g., for the exceptions monad, an important map is $\text{raise} \overset{\text{df}}{=} E \xrightarrow{\text{inr}} A + E$. 

Semantics of pure typed lambda calculus

- Pure typed lambda calculus can be interpreted into any Cartesian closed category $C$, e.g., $\textbf{Set}$.
- The interpretation is this:

\[
\begin{align*}
\llbracket K \rrbracket &= \text{df } \text{an object of } C \\
\llbracket A \times B \rrbracket &= \text{df } \llbracket A \rrbracket \times \llbracket B \rrbracket \\
\llbracket A \Rightarrow B \rrbracket &= \text{df } \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket \\
\llbracket C \rrbracket &= \text{df } \llbracket C_0 \rrbracket \times \ldots \times \llbracket C_{n-1} \rrbracket \\
\llbracket (x) x_i \rrbracket &= \text{df } \pi_i \\
\llbracket (x) \text{ let } x \leftarrow t \text{ in } u \rrbracket &= \text{df } \llbracket (x, x) u \rrbracket \circ \langle \text{id}, \llbracket (x) t \rrbracket \rangle \\
\llbracket (x) \text{ fst}(t) \rrbracket &= \text{df } \text{fst} \circ \llbracket (x) t \rrbracket \\
\llbracket (x) \text{ snd}(t) \rrbracket &= \text{df } \text{snd} \circ \llbracket (x) t \rrbracket \\
\llbracket (x) (t_0, t_1) \rrbracket &= \text{df } \langle \llbracket (x) t_0 \rrbracket, \llbracket (x) t_1 \rrbracket \rangle \\
\llbracket (x) \lambda x t \rrbracket &= \text{df } \Lambda(\llbracket (x, x) t \rrbracket) \\
\llbracket (x) t u \rrbracket &= \text{df } \text{ev} \circ \langle \llbracket (x) t \rrbracket, \llbracket (x) u \rrbracket \rangle 
\end{align*}
\]
This interpretation is sound: derivable typing judgements of the pure typed lambda calculus are valid, i.e.,

\[ \bar{x} : C \vdash t : A \text{ implies } \llbracket (\bar{x}) \ t \rrbracket : \llbracket C \rrbracket \rightarrow \llbracket A \rrbracket \]

and the same holds true about all derivable equalities.

This interpretation is also complete.
Pre-[Cartesian closed] structure of the Kleisli category of a strong monad

- Given a Cartesian (closed) category \( C \) and a \((1, \times)\) strong monad \( T \) on it, how much of that structure carries over to \( \text{Kl}(T) \)?

- We can manufacture “pre-products” in \( \text{Kl}(T) \) using the products of \( C \) and the strength \( s_l \) like this:

\[
\begin{align*}
A_0 \times^T A_1 & \overset{\text{df}}{=} A_0 \times A_1 \\
\text{fst}^T & \overset{\text{df}}{=} \eta \circ \text{fst} \\
\text{snd}^T & \overset{\text{df}}{=} \eta \circ \text{snd} \\
\langle k_0, k_1 \rangle^T & \overset{\text{df}}{=} \text{sl}^* \circ \text{sr} \circ \langle k_0, k_1 \rangle
\end{align*}
\]
\[
\begin{align*}
k & : C \to TA \\
\ell & : C \times A \to TB \\
\ell \bullet^T k & =_{df} \\
C \xrightarrow{\langle \text{id}_C, k \rangle} C \times TA & \xrightarrow{s_{\ell},A} T(C \times A) \xrightarrow{\ell^*} TB \\
\text{fst}^T & =_{df} A_0 \times A_1 \xrightarrow{\text{fst}} A_0 \xrightarrow{\eta} TA_0 \\
\text{snd}^T & =_{df} A_0 \times A_1 \xrightarrow{\text{snd}} A_1 \xrightarrow{\eta} TA_1 \\
\langle k_0, k_1 \rangle^T & =_{df} \\
C \xrightarrow{\langle k_0, k_1 \rangle} TA_0 \times TA_1 & \xrightarrow{s_{\ell},A} T(A_0 \times TA_1) \xrightarrow{s_{\ell},A_1^*} T(A_0 \times A_1)
\end{align*}
\]
The typing rules of products hold, but not all laws.

In particular, we do not get the $\beta$-law of products. Effects cannot be undone!

E.g., taking $T$ to be the exception monad defined by $TA = \text{df } A + E$ for some fixed $E$ we do not have $\text{snd}^T \circ^T \langle k_0, k_1 \rangle^T = k_1$.

Take $k_0 = \text{df } \text{raise} = \text{inr} : E \to TA$,

$k_1 = \text{df } \text{id}^T = \text{inl} : E \to TE$

Then $\langle k_0, k_1 \rangle^T = \text{inr} : E \to T(A \times E)$ and hence $\text{snd}^T \circ^T \langle k_0, k_1 \rangle^T = \text{inr} \neq \text{inl} = k_1$.

In fact, $\times^T$ is not even a bifunctor unless $T$ is commutative, although it is functorial in each argument separately. Effects do not commute in general!
“Pre-exponents” are defined from the exponents of $C$ by

\[
A \Rightarrow^T B =_{df} A \Rightarrow TB
\]

\[
ev^T =_{df} ev
\]

\[
\Lambda^T(k) =_{df} \eta \circ \Lambda(k)
\]

\[
ev_{A,B}^T =_{df} (A \Rightarrow TB) \times A \xrightarrow{ev_{A,TB}} TB
\]

\[
k : C \times A \rightarrow TB
\]

\[
\Lambda^T(k) =_{df} C \xrightarrow{\Lambda(k)} A \Rightarrow TB \xrightarrow{\eta} T(A \Rightarrow TB)
\]
It is not true that \( A \Rightarrow^T - : \mathbf{Kl}(T) \to \mathbf{Kl}(T) \) is right adjoint to \( - \times^T A : \mathbf{Kl}(T) \to \mathbf{Kl}(T) \).
So \( \Rightarrow^T \) is not a true exponent wrt. the preproduct \( \times^T \).

But \( A \Rightarrow^T - : \mathbf{Kl}(T) \to C \) is right adjoint to \( J(- \times A) : C \to \mathbf{Kl}(T) \):

\[
\begin{align*}
J(C \times A) & \to^T B \\
\hline
C \times A & \to TB \\
\hline
C & \to A \Rightarrow TB \\
\hline
C & \to A \Rightarrow^T B
\end{align*}
\]

We that say \( A \Rightarrow^T B \) is the *Kleisli exponent* of \( A, B \).

More about the pre-[Cartesian closed] structure of Kleisli categories in the story about arrows.
CoCartesian structure of the Kleisli category of a monad

- If $C$ is coCartesian (has coproducts), then $\mathbf{KI}(T)$ is coCartesian too, since $J$ as a left adjoint preserves colimits.
- Concretely, the coproduct on $\mathbf{KI}(T)$ is defined by

$$A_0 +^T A_1 =_{df} A_0 + A_1$$
$$\text{inl}^T =_{df} \eta \circ \text{inl}$$
$$\text{inr}^T =_{df} \eta \circ \text{inr}$$
$$[k_0, k_1]^T =_{df} [k_0, k_1]$$
Semantics of an effectful language

In the semantics of an effectful language, the semantic universe is the Kleisli category $\text{KI}(T)$ of the appropriate strong monad $T$ on a Cartesian closed base category $C$.

The pure fragment is interpreted into $\text{KI}(T)$ as if the language was pure, using the pre-[Cartesian closed] structure:

$$
\begin{align*}
\llbracket K \rrbracket^T &= \text{df an object of } \text{KI}(T) \\
\llbracket A \times B \rrbracket^T &= \text{df } \llbracket A \rrbracket^T \times^T \llbracket B \rrbracket^T \\
\llbracket A \Rightarrow B \rrbracket^T &= \text{df } \llbracket A \rrbracket^T \Rightarrow^T \llbracket B \rrbracket^T \\
\llbracket C \rrbracket^T &= \text{df } \llbracket C_0 \rrbracket^T \times^T \ldots \times^T \llbracket C_{n-1} \rrbracket^T 
\end{align*}
$$
\[
\begin{align*}
\llbracket (x) x_i \rrbracket^T &= \text{df} \quad \pi_i^T = \eta \circ \pi_j \\
\llbracket (x) \text{let } x \leftarrow t \text{ in } u \rrbracket^T &= \text{df} \quad \llbracket (x, x) u \rrbracket^T \circ^\top \langle \text{id}^T, \llbracket (x) t \rrbracket^T \rangle^T = (\llbracket (x, x) u \rrbracket^T)^* \circ \text{sl} \circ \langle \text{id}, \llbracket (x) t \rrbracket^T \rangle \\
\llbracket (x) \text{fst}(t) \rrbracket^T &= \text{df} \quad \text{fst}^T \circ^\top \llbracket (x)t \rrbracket^T = T \text{fst} \circ \llbracket (x)t \rrbracket^T \\
\llbracket (x) \text{snd}(t) \rrbracket^T &= \text{df} \quad \text{snd}^T \circ^\top \llbracket (x)t \rrbracket^T = T \text{snd} \circ \llbracket (x)t \rrbracket^T \\
\llbracket (x) (t_0, t_1) \rrbracket^T &= \text{df} \quad \langle \llbracket (x)t_0 \rrbracket^T, \llbracket (x)t_1 \rrbracket^T \rangle^T = \text{sl}^* \circ \text{sr} \circ \langle \llbracket (x)t_0 \rrbracket^T, \llbracket (x)t_1 \rrbracket^T \rangle \\
\llbracket (x) \lambda x t \rrbracket^T &= \text{df} \quad \Lambda^T(\llbracket (x, x) t \rrbracket^T) = \eta \circ \Lambda(\llbracket (x, x) t \rrbracket^T) \\
\llbracket (x) t u \rrbracket^T &= \text{df} \quad \text{ev}^T \circ^\top \langle \llbracket (x)t \rrbracket^T, \llbracket (x)u \rrbracket^T \rangle^T = \text{ev}^* \circ \text{sl}^* \circ \text{sr} \circ \langle \llbracket (x)t \rrbracket^T, \llbracket (x)u \rrbracket^T \rangle
\end{align*}
\]
As $\mathbf{KI}(T)$ is only pre-Cartesian closed, for this pure fragment, soundness of typing holds, i.e.,

$$
\forall x : C \vdash t : A \implies \llbracket (x) t \rrbracket^T : \llbracket C \rrbracket^T \rightarrow^T \llbracket A \rrbracket^T
$$

but not all equations of the pure typed lambda-calculus are validated.

In particular,

$$
\vdash t : A \implies \llbracket t \rrbracket^T : 1 \rightarrow^T \llbracket A \rrbracket^T
$$

so a closed term $t$ of a type $A$ denotes an element of $T[\llbracket A \rrbracket^T]$. 
Any effect-constructs must be interpreted specifically validating their desired typing rules and equations. E.g., for a language with exceptions we would use the exceptions monad and define

\[
\llbracket (x) \; \texttt{raise}(e) \rrbracket^T = \text{df} \quad \texttt{raise} \circ^T \llbracket (x) \; e \rrbracket^T
= \text{raise}^* \circ \llbracket (x) \; e \rrbracket^T
\]
Kleisli adjunction

- Given a monad $T$ on category $C$, in the opposite direction to that of $J : C \to \text{Kl}(T)$ there is a functor $U : \text{Kl}(T) \to C$ defined by
  - $UA = \text{df} \ TA$,
  - if $k : A \to^T B$, then $Uk = \text{df} \ TA \xrightarrow{k^*} TB$.
- $U$ is right adjoint to $J$.

$$
\begin{array}{cccc}
\text{KI}(T) & \xrightarrow{\text{J}} & C & \xrightarrow{\text{U}} & \text{KI}(T) \\
J & \downarrow & & \downarrow & \\
\text{C} & & & & \end{array}
$$

$$
\begin{array}{ccc}
JA \to^T B & \\
A \to TB & A \to UB \\
\end{array}
$$

- Importantly, $UJ = T$. Indeed,
  - $UJA = TA$,
  - if $f : A \to B$, then $UJf = (\eta_B \circ f)^* = Tf$.
- Moreover, the unit of the adjunction is $\eta$.
- $J \dashv U$ is the initial adjunction factorizing $T$ in this way.

There is also a final one, known as the Eilenberg-Moore
Kleisli categories

- In general one can define a *Kleisli category* on \( C \) to be
  - a category \( D \) with the same objects as \( C \)
  - together with an identity-on-objects functor \( J : C \to D \)
    with right adjoint \( U \).

- To give a monad is the same as to give Kleisli category.

- We already know that a monad \( T \) induces a Kleisli category \( D =_{df} KL(T) \).

- Given a Kleisli category \( D \), we obtain a monad by taking \( T =_{df} UJ \).
Monad maps

- A *monad map* between monads $T$, $S$ on a category $C$ is a natural transformation $\tau : T \rightarrow S$ satisfying

\[
\begin{array}{cccc}
A & \xrightarrow{\eta^T_A} & TA & \xrightarrow{\tau_A} & SA \\
\downarrow{\eta^S_A} & & \downarrow{\eta^S_A} & & \downarrow{\mu^S_A} \\
TA & \xrightarrow{\tau_A} & SA & & \\
\end{array}
\quad
\begin{array}{cccc}
TTA & \xrightarrow{\tau_{TA}} & STA & \xrightarrow{S_{\tau_A}} & SSA \\
\downarrow{\mu^T_A} & & \downarrow{\mu^S_A} & & \downarrow{\mu^S_A} \\
TA & \xrightarrow{\tau_A} & SA & & \\
\end{array}
\]

- Alternatively, a map between two monads (Kleisli triples) $T$, $S$ is, for any object $A$, a map $\tau_A : TA \rightarrow SA$ satisfying
  - $\tau_A \circ \eta^T_A = \eta^S_A$,
  - if $k : A \rightarrow TB$, then $\tau_B \circ k^*T = (\tau_B \circ k)^*S \circ \tau_A$.
  (No explicit naturality condition on $\tau$.)

The two definitions are equivalent.

- Monads on $C$ and maps between them form a category $\text{Monad}(C)$. 
There is a bijection between monad maps $\tau$ between $T$, $S$ and functors $V : \text{KI}(T) \to \text{KI}(S)$ satisfying $V J^T = J^S$.

Given $\tau$, one defines $V$ by
- $VA \overset{\text{df}}{=} A$,
- if $k : A \to TB$, then $Vk \overset{\text{df}}{=} A \xrightarrow{k} TB \xrightarrow{\tau_B} SB$.

Given $V$, one defines $\tau$ by
- $\tau_A \overset{\text{df}}{=} V(TA \xrightarrow{\text{id}_TA} TA) : TA \to^S A$. 

- Monad maps vs. functors between Kleisli categories