Outline

- Monads and why they matter for a working functional programmer
- Combining monads: monad transformers, distributive laws, the coproduct of monads
- Finer and coarser: Lawvere theories and arrows
- Comonadic notions of computation: dataflow notions of computation, 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:

\[
\begin{align*}
T \eta_A &\quad \eta_{TA} \\
\downarrow T \eta_A &\quad \downarrow \mu_A \\
TTA &\quad \mu_T \\
\mu_A &\quad \mu_A
\end{align*}
\]

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 $T : |C| \to |C|$, for any object $A$, a map $\eta_A : A \to TA$, for any map $k : A \to TB$, a map $k^* : TA \to TB$ (the Kleisli extension operation)

satisfying these conditions:
  - if $k : A \to TB$, then $k^* \circ \eta_A = k$,
  - $\eta_A^* = id_{TA}$,
  - if $k : A \to TB$, $\ell : B \to TC$, 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 $k^* =_{df} TA \xrightarrow{Tk} TTB \xrightarrow{\mu_B} TB$.

- Given $T$ (on objects only), $\eta$ and $-^*$, one defines
  
  if $f : A \to B$, then
  
  $Tf =_{df} ( A \xrightarrow{f} B \xrightarrow{\eta_B} TB )^* : TA \to TB$,

  $\mu_A =_{df} ( TA \xrightarrow{id_{TA}} TA )^* : TTA \to TA$. 
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 $TB$,
  - $\text{id}_A^T = \text{df} \ A \xrightarrow{\eta_A} TA$,
  - 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 $\mathcal{C}$ as the category of pure functions and of $TA$ as the type of effectful computations of values of a type $A$.
- $\text{KI}(T)$ 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.
Kleisli adjunction

- In the opposite direction there is a functor $U : \text{Kl}(T) \to C$ defined by
  - $UA \overset{\text{df}}{=} TA$,
  - if $k : A \to^T B$, then $Uk \overset{\text{df}}{=} TA \xrightarrow{k^*} TB$.
- $J$ is left adjoint to $U$.

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

- 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 adjunction.
Examples

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

- Output monad:
  - $TA = \text{df} \ A \times E$ 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,
  - $\eta_A = \text{df} \ A \xrightarrow{\text{ur}} A \times 1 \xrightarrow{\text{id} \times e} A \times E$,
  - $\mu_A = \text{df} \ (A \times E) \times E \xrightarrow{a} A \times (E \times E) \xrightarrow{\text{id} \times m} A \times E$,
  - if $k : A \to B \times E$, then
    - $k^* = \text{df} \ A \times E \xrightarrow{k \times \text{id}} (B \times E) \times E \xrightarrow{a} B \times (E \times E) \xrightarrow{\text{id} \times m} B \times E$. 
Reader monad:

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

Side-effect monad:

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

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

satisfying

\[
\begin{align*}
I \otimes FA & \xrightarrow{sl_{I,A}} F(I \otimes A) \\
ul_{FA} & \downarrow \quad Ful_{A} \\
FA & \xrightarrow{\text{id}} FA \\
(A \otimes B) \otimes FC & \xrightarrow{sl_{A\otimes B,C}} F((A \otimes B) \otimes C) \\
a_{A,B,FC} & \downarrow \quad Fa_{A,B,C} \\
A \otimes (B \otimes FC) & \xrightarrow{id_{A} \otimes sl_{B,C}} A \otimes F(B \otimes C) \xrightarrow{sl_{A,B\otimes C}} F(A \otimes (B \otimes C))
\end{align*}
\]
A strong natural transformation between two strong functors \((F, sl), (G, sl')\) is a natural transformation \(\tau : F \rightarrow G\) satisfying:

\[
A \otimes FB \xrightarrow{sl_{A,B}} F(A \otimes B) \\
\downarrow \text{id}_{A \otimes \tau_B} \downarrow \tau_{A \otimes B} \\
A \otimes GB \xrightarrow{sl'_{A,B}} G(A \otimes B)
\]
**Strong monads**

A *strong monad* on a monoidal category \((C, I, \otimes)\) is a monad \((T, \eta, \mu)\) together with a strength \(s_l\) for \(T\) for which \(\eta\) and \(\mu\) are strong, i.e., satisfy

\[
\begin{align*}
A \otimes B \cong A \otimes B
\end{align*}
\]

\[
\begin{align*}
&\quad \text{id}_{A \otimes B} \downarrow \quad \text{id}_{A \otimes B} \\
A \otimes TB \quad &\xrightarrow{s_l_{A,B}} \quad T(A \otimes B)
\end{align*}
\]

\[
\begin{align*}
A \otimes TTB \quad &\xrightarrow{s_l_{A,TB}} \quad T(A \otimes TB) \quad \xrightarrow{T s_l_{A,B}} \quad TT(A \otimes B)
\end{align*}
\]

\[
\begin{align*}
&\quad \text{id}_{A \otimes \mu B} \downarrow \quad \text{id}_{A \otimes \mu B} \\
A \otimes TB \quad &\xrightarrow{s_l_{A,B}} \quad T(A \otimes B)
\end{align*}
\]

(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

\[
\begin{align*}
    sr_{A,B} &= \text{df } FA \otimes B \xrightarrow{c_{FA,B}} B \otimes FA \xrightarrow{sl_{B,A}} F(B \otimes A) \xrightarrow{F_{CB,A}} F(A \otimes B)
\end{align*}
\]

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

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

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

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

- A functorially strong functor on a monoidal closed category \((C, I, \otimes, \rightarrow)\) is an endofunctor \(F\) on \(C\) with a natural transformation \(\text{fs}_{A,B}: A \rightarrow B \rightarrow FA \rightarrow FB\) 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 \(\text{fs}\), one defines \(\text{sl}\) by

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

- Given \(\text{sl}\), one defines \(\text{fs}\) by

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

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

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

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

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 $\textbf{Kl}(T)$?

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

  $A_0 \times^T A_1 =_{df} A_0 \times A_1$

  $\text{fst}^T =_{df} \eta \circ \text{fst}$

  $\text{snd}^T =_{df} \eta \circ \text{snd}$

  $\langle k_0, k_1 \rangle^T =_{df} \text{sl}^* \circ \text{sr} \circ \langle k_0, k_1 \rangle$
\[
\begin{align*}
&k : C \to TA \\
&\ell : C \times A \to TB \\
\ell \bullet^T k &= \text{df} \\
&\langle \text{id}_C, k \rangle^C \to C \times TA \xrightarrow{\text{sl}_{C,A}} T(C \times A) \xrightarrow{\ell^*} TB \\
\text{fst}^T &= \text{df} \quad A_0 \times A_1 \xrightarrow{\text{fst}} A_0 \xrightarrow{\eta} TA_0 \\
\text{snd}^T &= \text{df} \quad A_0 \times A_1 \xrightarrow{\text{snd}} A_1 \xrightarrow{\eta} TA_1 \\
&k_0 : C \to TA_0 \\
&k_1 : C \to TA_1 \\
\langle k_0, k_1 \rangle^T &= \text{df} \\
&C \langle k_0, k_1 \rangle \xrightarrow{T\text{f}_0} TA_0 \times TA_1 \xrightarrow{\text{sr}_{A_0,TA_1}} T(A_0 \times TA_1) \xrightarrow{\text{sl}_{A_0,A_1}^*} T(A_0 \times A_1)
\end{align*}
\]
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 $TA \triangleq df A + E$ for some fixed $E$ we do not have $\text{snd}^T \circ^T \langle k_0, k_1 \rangle^T = k_1$.

Take $k_0 \triangleq df \text{raise} = \text{inr} : E \to TA,$ $k_1 \triangleq df \text{id}^T = \text{inl} : E \to TE$
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

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

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

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

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

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

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

But $A \Rightarrow^T - : \text{KI}(T) \to C$ is right adjoint to $J(- \times A) : C \to \text{KI}(T)$:

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

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

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

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

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

$$[[K]]^T = \text{df} \quad \text{an object of } \text{Kl}(T)$$

$$[[A \times B]]^T = \text{df} \quad \text{that object of } \mathcal{C}$$

$$[[A \times B]]^T = \text{df} \quad [A]^T \times^T [B]^T$$

$$[[A \Rightarrow B]]^T = \text{df} \quad [A]^T \Rightarrow^T [B]^T$$

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

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

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

\[
\llbracket (x) \text{ raise}(e) \rrbracket^T = \text{df} \quad \text{raise} \circ^T \llbracket (x) e \rrbracket^T \\
= \text{raise}^* \circ \llbracket (x) e \rrbracket^T
\]
Monad maps

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

$$
\eta_A^T \downarrow \quad \eta_A^S \\
A \quad \quad A \\
\eta_A^T \\
TA \quad \tau_A \quad SA \\
\mu_A^T \downarrow \\
S TA \quad \tau \quad SSA \\
\mu_A^S \\
T TA \quad \tau_{TA} \quad STA \quad \mu_{TA} \quad SSA
$$

- Alternatively, a map between two monads (Kleisli triples) $T, S$ is, for any object $A$, a map $\tau_A : TA \rightarrow SA$ satisfying
  - $\tau_A \circ \eta_A^T = \eta_A^S$,
  - 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 $\text{Monad}(C)$. 
Monad maps vs. functors between Kleisli categories

- There is a bijection between monad maps $\tau$ between $T$, $S$ and functors $V : \mathbf{KI}(T) \to \mathbf{KI}(S)$ satisfying $VJ^T = J^S$.
- 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
  - $\tau_A \overset{\text{df}}{=} V(TA \xrightarrow{id_T A} TA) : TA \to^S A$. 