Containers for Effects and Contexts

Tarmo Uustalu, Institute of Cybernetics, Tallinn

University of Oxford, 6–10 July 2015

This course

- We will think about computational effects and contexts as modelled with monads, comonads and related machinery.

- We will primarily be interested in questions like: Where do they come from? How to generate them? How many are they?
  And also: How to arrive at answers to such questions with as little work as possible?
- In other words, we will amuse ourselves with the combinatorics of monads etc.

- The main tool: Containers (possibly quotient containers). But not today.
- Today's ambition: Monads, monad maps and distributive laws.

Useful prior knowledge

- This is not strictly needed, but will help.
- Basics of functional programming and the use of monads (and perhaps idioms, comonads) in functional programming.
- From category theory:
  - functors, natural transformations
  - adjunctions
  - symmetric monoidal (closed) categories
  - Cartesian (closed) categories, coproducts
  - initial algebra, final coalgebra of a functor
  - . . . :-(
- All examples however will be for **Set**. :-)
- (But many generalize to any Cartesian (closed) or monoidal (closed) category.)

Monads

Monads

- A *monad* on a category $\mathcal{C}$ is given by a
    - a functor $T : \mathcal{C} \to \mathcal{C}$,
    - a natural transformation $\eta : \mathrm{Id}_{\mathcal{C}} \dot{\to} T$ (the *unit*),
    - a natural transformation $\mu : T \cdot T \dot{\to} T$ (the *multiplication*)

  such that

$$
\begin{tikzcd}
TA \arrow[r, "T\eta_A"] \arrow[d, "\eta_{TA}"'] \arrow[dr, equal] & T(TA) \arrow[d, "\mu_A"] \\
T(TA) \arrow[r, "\mu_A"'] & TA
\end{tikzcd}
\qquad
\begin{tikzcd}
T(T(TA)) \arrow[r, "T\mu_A"] \arrow[d, "\mu_{TA}"'] & T(TA) \arrow[d, "\mu_A"] \\
T(TA) \arrow[r, "\mu_A"'] & TA
\end{tikzcd}
$$

- This definition says that monads are monoids in the monoidal category $([\mathcal{C}, \mathcal{C}], \mathrm{Id}_{\mathcal{C}}, \cdot)$.

An alternative formulation: Kleisli triples

- A more FP-friendly formulation is this.
- A *Kleisli triple* is given by
  - an object mapping $T : |\mathcal{C}| \to |\mathcal{C}|$,
  - for any object A, a map $\eta_A : A \to TA$,
  - for any map $k : A \to TB$, a map $k^\star : TA \to TB$ (the *Kleisli extension* operation)

  such that
  - if $k : A \to TB$, then $k^\star \circ \eta_A = k$,
  - $\eta_A^\star = \mathrm{id}_{TA}$,
  - if $k : A \to TB$, $\ell : B \to TC$, then $(\ell^\star \circ k)^\star = \ell^\star \circ k^\star : TA \to TC$.
- (Notice there are no explicit functoriality and naturality conditions.)

Monads = Kleisli triples

- There is a bijection between monads and Kleisli triples.
- Given T, η, μ, one defines
  - if $k : A \to TB$, then $k^\star = \; TA \xrightarrow{Tk} T(TB) \xrightarrow{\mu_B} TB$.
- Given T (on objects only), η and $-^\star$, one defines
  - if $f : A \to B$, then
    $$Tf = \left(A \xrightarrow{f} B \xrightarrow{\eta_B} TB \right)^\star : TA \to TB,$$
  - $$\mu_A = \left(TA \xrightarrow{\mathrm{id}_{TA}} TA \right)^\star : T(TA) \to TA.$$

Kleisli category of a monad

- A monad T on a category $\mathcal{C}$ induces a category $\mathbf{Kl}(T)$ called the *Kleisli category* of T defined by
  - an object is an object of $\mathcal{C}$,
  - a map of from A to B is a map of $\mathcal{C}$ from A to TB,
  - $\mathrm{id}_A^T = A \xrightarrow{\eta_A} TA$,
  - if $k : A \to^T B$, $\ell : B \to^T C$, then

$$\begin{tikzcd} \ell \circ^T k = A \arrow[r, "k"] & TB \arrow[r, "T\ell"] \arrow[rr, bend right, "\ell^\star"'] & T(TC) \arrow[r, "\mu_C"] & TC \end{tikzcd}$$

- From $\mathcal{C}$ there is an identity-on-objects *inclusion* functor J to $\mathbf{Kl}(T)$, defined on maps by
  - if $f : A \to B$, then

$$Jf = A \xrightarrow{f} B \xrightarrow{\eta_B} TB = A \xrightarrow{\eta_A} TA \xrightarrow{Tf} TB .$$

Monad algebras

- An *algebra* of a monad (T, η, μ) is an object A with a map $a : TA \to A$ such that

$$\begin{tikzcd} A \arrow[d, "\eta_A"'] \arrow[dr, equal] & \\ TA \arrow[r, "a"'] & A \end{tikzcd} \qquad \begin{tikzcd} T(TA) \arrow[r, "Ta"] \arrow[d, "\mu_A"'] & TA \arrow[d, "a"] \\ TA \arrow[r, "a"'] & A \end{tikzcd}$$

- A *map* between two algebras (A, a) and (B, b) is a map h such that

$$\begin{tikzcd} TA \arrow[r, "Th"] \arrow[d, "a"'] & TB \arrow[d, "b"] \\ A \arrow[r, "h"'] & B \end{tikzcd}$$

- The algebras of the monad and maps between them form a category $\mathbf{EM}(T)$ with an obvious forgetful functor $U : \mathbf{EM}(T) \to \mathcal{C}$.

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.
- $\eta_A : A \to TA$ 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.
- $\mu_A : T(TA) \to TA$ flattens an effectful computation of an effectful computation.
- $k^\star : TA \to TB$ is an effectful function $k : A \to TB$ extended into one that can input an effectful computation.
- An algebra $(A, a : TA \to A)$ serves as a recipe for handling the effects in computations of values of type A.

Kleisli adjunction

- In the opposite direction of $J : \mathcal{C} \to \mathbf{Kl}(T)$ there is a functor $R : \mathbf{Kl}(T) \to \mathcal{C}$ defined by
  - $RA = TA$,
  - if $k : A \to^T B$, then $Rk = \; TA \xrightarrow{k^\star} TB$.
- R is right adjoint to J.

$$\begin{tikzcd} \mathbf{Kl}(T) \arrow[d, bend left, "R"] \arrow[d, phantom, "\dashv"] \\ \mathcal{C} \arrow[u, bend left, "J"] \end{tikzcd} \qquad \frac{\overbrace{A}^{JA} \to^T B}{A \to \underbrace{TB}_{RB}}$$

- Importantly, $R \cdot J = T$. Indeed,
  - $R(JA) = TA$,
  - if $f : A \to B$, then $R(Jf) = (\eta_B \circ f)^\star = Tf$.
- Moreover, the unit of the adjunction is η.
- $J \dashv R$ is the initial adjunction factorizing T in this way.

Eilenberg-Moore adjunction

- In the opposite direction of $U : \mathbf{EM}(T) \to \mathcal{C}$ there is a functor $L : \mathcal{C} \to \mathbf{EM}(T)$ defined by
  - $LA = (TA, \mu_A)$,
  - if $f : A \to B$, then $Lf = Tf : (TA, \mu_A) \to (TB, \mu_B)$.
- L is left adjoint to U.

$$
\begin{tikzcd}
\mathbf{EM}(T) \arrow[d, bend left, "U"] \arrow[d, phantom, "\dashv"] \\
\mathcal{C} \arrow[u, bend left, "L"]
\end{tikzcd}
\qquad
\begin{array}{c}
\overbrace{(TA, \mu_A)}^{LA} \to (B, b) \\
\hline\hline
A \to \underbrace{B}_{U(B,b)}
\end{array}
$$

- $U \cdot L = T$. Indeed,
  - $U(LA) = U(TA, \mu_A) = TA$,
  - if $f : A \to B$, then $U(Lf) = U(Tf) = Tf$.
- The unit of the adjunction is η.
- $L \dashv U$ is the final adjunction factorizing T.

Exceptions monads

- The functor:
  - $TA = E + A$ where E is some set (of exceptions)
- The monad structure:
  - $\eta_A\, x = \mathsf{inr}\, x$,
  - $\mu_A\, (\mathsf{inl}\, e) = \mathsf{inl}\, e$,
    $\mu_A\, (\mathsf{inr}\, (\mathsf{inl}\, e)) = \mathsf{inl}\, e$,
    $\mu_A\, (\mathsf{inr}\, (\mathsf{inr}\, x)) = \mathsf{inr}\, x$.
- This is the only monad structure on this functor.

- (This example generalizes to any coCartesian category, in fact to any monoidal category with a given monoid.
  In a coCartesian category, any object E carries exactly one monoid structure defined by $\mathsf{o} = ?_E : 0 \to E$ and $\oplus = \nabla_E : E + E \to E$.)

Reader monads

- The functor:
  - $TA = S \Rightarrow A$ where S is a set (of readable states)
- The monad structure:
  - $\eta_A\, x = \lambda s.\, x$,
  - $\mu_A\, f = \lambda s.\, f\, s\, s$.
- This is the only monad structure on this functor.

- (This example generalizes to any monoidal closed category with a given comonoid. In a Cartesian closed category, any object S comes with a unique comonoid structure given by $!_S : S \to 1$, $\Delta_S : S \to S \times S$.)

Writer monads

- We are interested in this functor:
  - $TA = P \times A$ where P is a set (of updates)
- The possible monad structures are:
  - $\eta_A\, x = (\mathrm{o}, x)$,
  - $\mu_A\, (p, (p', x)) = (p \oplus p', x)$
    where $(\mathrm{o}, \oplus)$ is a monoid structure on P (trivial update, composition of updates)
- Monad structures on this functor are in a bijection with monoid structures on P.

- (This example generalizes to any monoidal category with a given monoid.)

State monads

- The monad:
  - $T\,A = S \Rightarrow S \times A$ where S is a set (of readable/overwritable states),
  - $\eta_A\,x = \lambda s.\,(s, x)$
  - $\mu_A\,f = \lambda s.\,\mathrm{let}\ (s', g) = f\,s\ \mathrm{in}\ g\,(s', x)$

- (This example works in any monoidal closed category.)

List monad and variations

- The list monad:
  - $TA = \mathsf{List}\,A$,
  - $\eta_A\,x = [x]$,
  - $\mu_A\,xss = \mathsf{concat}\,xss$.
- Some variations:
  - $TA = \{xs : A^* \mid xs \text{ is square-free}\}$
  - $TA = \{xs : A^* \mid xs \text{ is duplicate-free}\}$
  - $TA = 1 + A \times A$
  - $TA = \mathcal{M}_{\mathrm{f}}\,A$
  - $TA = \mathcal{P}_{\mathrm{f}}\,A$
  - non-empty versions of the above
- Can you characterize the algebras of these monads?

Monad maps

Monad maps

- A *monad map* between monads T, T' on a category $\mathcal{C}$ is a natural transformation $\tau : T \stackrel{\cdot}{\to} T'$ satisfying

$$
\begin{tikzcd}
A \arrow[r, equal] \arrow[d, "\eta_A"'] & A \arrow[d, "\eta'_A"] \\
TA \arrow[r, "\tau_A"'] & T'A
\end{tikzcd}
\qquad
\begin{tikzcd}
T(TA) \arrow[r, "\tau_{TA}"] \arrow[d, "\mu_A"'] & T'(TA) \arrow[r, "T'\tau_A"] & T'(T'A) \arrow[d, "\mu'_A"] \\
TA \arrow[rr, "\tau_A"'] & & T'A
\end{tikzcd}
$$

- Monads on $\mathcal{C}$ and maps between them form a category $\mathbf{Monad}(\mathcal{C})$.
- Monad maps are monoid maps in the monoidal category $([\mathcal{C},\mathcal{C}], \mathrm{Id}_{\mathcal{C}}, \cdot)$ and the category of monads is the category of monoids in $([\mathcal{C},\mathcal{C}], \mathrm{Id}_{\mathcal{C}}, \cdot)$.

Kleisli triple maps

- A map between two Kleisli triples T, T' is, for any object A, a map $\tau_A : TA \to T'A$ such that
  - $\tau_A \circ \eta_A = \eta'_A$,
  - if $k : A \to TB$, then $\tau_B \circ k^\star = (\tau_B \circ k)^{\star\prime} \circ \tau_A$.
- (No explicit naturality condition on τ!)
- Kleisli triples on $\mathcal{C}$ and maps between them form a category that is isomorphic to **Monad**$(\mathcal{C})$.

Monad maps vs. functors between Kleisli categories

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

$$\begin{tikzcd} \mathbf{Kl}(T) \arrow[rr, "V"] & & \mathbf{Kl}(T') \\ & \mathcal{C} \arrow[ul, "J"] \arrow[ur, "J'"'] & \end{tikzcd}$$

- This is defined by
  - $VA = A$,
  - if $k : A \to TB$, then $Vk = A \xrightarrow{k} TB \xrightarrow{\tau_B} T'B$.

and

  - $\tau_A = V(TA \xrightarrow{\mathrm{id}_{TA}} {}^{T}A) : TA \to^{T'} A$.

Monad maps vs. functors between E-M categories

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

$$\begin{tikzcd} \mathbf{EM}(T') \arrow[rr, "V"] \arrow[dr, "U'"'] & & \mathbf{EM}(T) \arrow[dl, "U"] \\ & \mathcal{C} & \end{tikzcd}$$

  (Note the reversed direction.)

- This is defined by
  - $V(A, a) = (A, a \circ \tau_A)$,
  - if $h : (A, a) \to (B, b)$, then $Vh = h : (A, a \circ \tau_A) \to (B, b \circ \tau_B)$.

  and

  - $\tau_A = \text{let } (T'A, a) \leftarrow V(T'A, \mu'_A) \text{ in } a \circ T\eta'_A$.

Examples: Exceptions, reader, writer monads

- Monad maps between the exception monads for sets E, E' are in a bijection with pairs of an element of $E' + 1$ and a function between E and E'.
  (Why?)

- 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 $(P, \mathrm{o}, \oplus)$ and $(P', \mathrm{o}', \oplus')$ are in a bijection with homomorphisms between these monoids.

Examples: From exceptions to writer or vice versa

- There is no monad map τ from the exception monad for a set E and the writer monad for a monoid $(P, \mathrm{o}, \oplus)$ (unless $E = 0$).
  There is not even a natural transformation between the underlying functors: it is impossible to have a map $\tau_0 : 0 + E \to P \times 0$.

- Monad maps τ from the writer monad for $(P, \mathrm{o}, \oplus)$ to the exception monad for E are in a bijection between monoid homomorphisms between $(P, \mathrm{o}, \oplus)$ and the free monoid on the left zero semigroup on E. (Can you simplify this condition further?)
  They can be written as

$$\tau_X = P \times X \longrightarrow (E+1) \times X \longrightarrow E \times X + 1 \times X \longrightarrow E + X$$

Examples: Reader and state monads

- The monad maps between the state monads for S and C are in a bijection with *lenses*, i.e., pairs of functions $lkp : C \to S$, $upd : C \times S \to C$ such that
  - $lkp\,(upd\,(c, s)) = s$,
  - $upd\,(c, lkp\,c)) = c$,
  - $upd\,(upd\,(c, s), s') = upd\,(c, s')$.

- Can you characterize the monad maps from the reader monad for S to the state monad for C? The other way around? (Be careful here!)

Examples: Nonempty lists and powerset

- How many monad maps are there from the nonempty list monad to itself?
- Answer: 4, viz. the identity map, reverse, take only the first element, take only the last element.
- Why does taking the 2nd element not qualify? Or taking the two first elements? (These are natural transformations, but. . .)

- How many monad maps are there from the nonempty list monad to the nonempty powerset monad? The other way around?

Compatible compositions of monads

Compatible compositions of monads

- A *compatible composition* of two monads (T_0, η_0, μ_0), (T_1, η_1, μ_1) is a monad structure (η, μ) on $T = T_0 \cdot T_1$ satisfying

$$
\begin{tikzcd}
\mathrm{Id} \arrow[r, "\eta"'] \arrow[r, bend left, "\eta_0 \cdot \eta_1"] & T_0 \cdot T_1
\end{tikzcd}
\qquad
\begin{tikzcd}
T_0 \cdot T_0 \arrow[r, "\mu_0"] \arrow[d, "T_0 \cdot \eta_1 \cdot T_0 \cdot \eta_1"'] & T_0 \arrow[d, "T_0 \cdot \eta_1"] \\
T_0 \cdot T_1 \cdot T_0 \cdot T_1 \arrow[r, "\mu"'] & T_0 \cdot T_1
\end{tikzcd}
$$

$$
\begin{tikzcd}
T_1 \cdot T_1 \arrow[r, "\mu_1"] \arrow[d, "\eta_0 \cdot T_1 \cdot \eta_0 \cdot T_1"'] & T_1 \arrow[d, "\eta_0 \cdot T_1"] \\
T_0 \cdot T_1 \cdot T_0 \cdot T_1 \arrow[r, "\mu"'] & T_0 \cdot T_1
\end{tikzcd}
\qquad
\begin{tikzcd}
 & T_0 \cdot T_1 \arrow[dl, "T_0 \cdot \eta_1 \cdot \eta_0 \cdot T_1"'] \arrow[dr, equal] & \\
T_0 \cdot T_1 \cdot T_0 \cdot T_1 \arrow[rr, "\mu"'] & & T_0 \cdot T_1
\end{tikzcd}
$$

- Conditions 1-3 say just that $T_0 \cdot \eta_1$ and $\eta_0 \cdot T_1$ are monad morphisms between (T_0, η_0, μ_0) resp. (T_1, η_1, μ_1) and (T, η, μ).
  Condition 1 fixes that $\eta = \eta_0 \cdot \eta_1$; so the only freedom is about μ.

Distributive laws of monads

- A *distributive law* of a monad (T_1, η_1, μ_1) over (T_0, η_0, μ_0) is a natural transformation $\theta : T_1 \cdot T_0 \to T_0 \cdot T_1$ such that

$$\begin{tikzcd} & T_1 \arrow[dl, "T_1 \cdot \eta_0"'] \arrow[dr, "\eta_0 \cdot T_1"] & \\ T_1 \cdot T_0 \arrow[rr, "\theta"'] & & T_0 \cdot T_1 \end{tikzcd} \qquad \begin{tikzcd} T_1 \cdot T_0 \cdot T_0 \arrow[r, "\theta \cdot T_0"] \arrow[d, "T_1 \cdot \mu_0"'] & T_0 \cdot T_1 \cdot T_0 \arrow[r, "T_0 \cdot \theta"] & T_0 \cdot T_0 \cdot T_1 \arrow[d, "\mu_0 \cdot T_1"] \\ T_1 \cdot T_0 \arrow[rr, "\theta"'] & & T_0 \cdot T_1 \end{tikzcd}$$

$$\begin{tikzcd} & T_0 \arrow[dl, "\eta_1 \cdot T_0"'] \arrow[dr, "T_0 \cdot \eta_1"] & \\ T_1 \cdot T_0 \arrow[rr, "\theta"'] & & T_0 \cdot T_1 \end{tikzcd} \qquad \begin{tikzcd} T_1 \cdot T_1 \cdot T_0 \arrow[r, "T_1 \cdot \theta"] \arrow[d, "\mu_1 \cdot T_0"'] & T_1 \cdot T_0 \cdot T_1 \arrow[r, "\theta \cdot T_1"] & T_0 \cdot T_1 \cdot T_1 \arrow[d, "T_0 \cdot \mu_1"] \\ T_1 \cdot T_0 \arrow[rr, "\theta"'] & & T_0 \cdot T_1 \end{tikzcd}$$

Compatible compositions = distributive laws

- Compatible compositions of (T_0, η_0, μ_0), (T_1, η_1, μ_1) are in a bijection with distributive laws of (T_1, η_1, μ_1) over (T_0, η_0, μ_0).
- Given μ, one recovers θ by

$$\theta = \; T_1 \cdot T_0 \xrightarrow{\eta_0 \cdot T_1 \cdot T_0 \cdot \eta_1} T_0 \cdot T_1 \cdot T_0 \cdot T_1 \xrightarrow{\mu} T_0 \cdot T_1$$

- Given θ, μ is defined by

$$\mu = \; T_0 \cdot T_1 \cdot T_0 \cdot T_1 \xrightarrow{T_0 \cdot \theta \cdot T_1} T_0 \cdot T_0 \cdot T_1 \cdot T_1 \xrightarrow{\mu_0 \cdot \mu_1} T_0 \cdot T_1$$

Algebras of compatible compositions

- Given a distributive law θ, a θ-*pair of algebras* is given by a set A with a (T_0, η_0, μ_0)-algebra structure (A, a_0) and a (T_1, η_1, μ_1)-algebra structure (A, a_1) such that

$$\begin{tikzcd} T_1 A \arrow[d, "T_1 a_0"'] & A \arrow[l, "a_1"'] \arrow[r, "a_0"] & T_0 A \arrow[d, "T_0 a_1"] \\ T_1(T_0 A) \arrow[rr, "\theta_A"] & & T_0(T_1 A) \end{tikzcd}$$

- Such pairs of algebras are in a bijection with (T, η, μ)-algebras.
- Given a_0, a_1, one constructs a as
  - $a = T_0(T_1 A) \xrightarrow{T_0 a_1} T_0 A \xrightarrow{a_0} A.$
- Given a, a_0 and a_1 are defined by
  - $a_0 = T_0 A \xrightarrow{T_0 \eta_{1A}} T_0(T_1 A) \xrightarrow{a} A,$
  - $a_1 = T_1 A \xrightarrow{\eta_0\, T_1 A} T_0(T_1 A) \xrightarrow{a} A.$

Any monad and an exceptions monad

- The exceptions monad for E distributes in a unique way over any monad (T_0, η_0, μ_0).
- $\theta : E + T_0 A \to T_0(E + A)$
  $\theta_A\,(\mathsf{inl}\,e) = \eta_0\,(\mathsf{inl}\,e)$,
  $\theta_A\,(\mathsf{inr}\,c) = T_0\,\mathsf{inr}$
- So we have a unique monad structure on $TA = T_0(E + A)$ that is compatible with (T_0, η_0, μ_0).
- (This generalizes to any coCartesian category, also to any monoidal category with a comonoid.)

Any monad and a writer monad

- There is a unique distributive law of the writer monad for $(P, \mathrm{o}, \oplus)$ over any monad (T_0, η_0, μ_0).
- $\theta : P \times T_0 A \to T_0(P \times A)$
  $\theta_A\,(p, c) = T_0(\lambda x.\,(p, x))\,c$.
  (θ is nothing but the unique strength of T_0!)
- So monad structures on $TA = T_0(P \times A)$ compatible with (T_0, η_0, μ_0) are in a bijection with monoid structures on P.

- (This generalizes to any Cartesian category and any monoidal category in the form of a bijection between strengths and distributive laws.)

Monoid actions

- A *right action* of a monoid $(P, \mathsf{o}, \oplus)$ on a set S is a map $\downarrow : S \times P \to S$ satisfying

$$s \downarrow \mathsf{o} = s$$
$$s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$$

Reader and writer monads

- Distributive laws of the writer monad for $(P, \mathsf{o}, \oplus)$ over the reader monad for S are in a bijective correspondence with right actions of $(P, \mathsf{o}, \oplus)$ on S.
- The compatible composition of the two monads determined by a right action $\downarrow$ is

$$
\begin{aligned}
T\,A &= S \Rightarrow P \times A \\
\eta\, x &= \lambda s.\, (\mathsf{o}, x) \\
\mu\, f &= \lambda s.\, \mathrm{let}\ \ \begin{array}[t]{l} (p, g) = f\ s \\ (p', x) = g\,(s \downarrow p) \end{array} \\
&\qquad\ \ \mathrm{in}\ (p \oplus p', x)
\end{aligned}
$$

—the update monad for S, $(P, \mathsf{o}, \oplus)$, $\downarrow$.

State logging

- Take S to be some set (of states).
- Take $P = \text{List}\, S$, $\mathsf{o} = []$, $\oplus = +\!\!+$ (state logs).
- $s \downarrow [] = s$
  $s \downarrow (s' :: ss) = s' \downarrow ss$

  (so $s \downarrow ss$ is the last element of $s :: ss$)

Reading a stack and popping

- Take $S = \mathsf{List}\, E$ (states of a stack of elements drawn from a set E).
- Take $P = \mathsf{Nat}$, $\mathsf{o} = 0$, $\oplus = +$ (possible numbers of elements to pop).
- $es \downarrow n = removelast\, n\, es$.

Reading a stack and pushing

- Take again $S = \mathsf{List}\, E$ (states of a stack of elements drawn from a set E).
- Take $P = \mathsf{List}\, E$, $\mathrm{o} = []$, $\oplus = +\!\!+$ (lists of elements to push on the stack).
- $es \downarrow es' = es +\!\!+ es'$.
- (So here we choose $(S, \downarrow)$ to be the initial $(P, \mathrm{o}, \oplus)$-set—which is always a possibility.)

Matching pairs of monoid actions

- A *matching pair of actions* of two monoids $(P_0, o_0, \oplus_0)$ and $(P_1, o_1, \oplus_1)$ on each other is pair of maps $\searrow : P_1 \times P_0 \to P_0$ and $\swarrow : P_1 \times P_0 \to P_1$ such that

$$\begin{aligned}
o_1 \searrow p_0 &= p_0 \\
(p_1 \oplus_1 p_1') \searrow p_0 &= p_1 \searrow (p_1' \searrow p_0) \\
p_1 \searrow o_0 &= o_0 \\
p_1 \searrow (p_0 \oplus_0 p_0') &= (p_1 \searrow p_0) \oplus_0 ((p_1 \swarrow p_0) \searrow p_0')
\end{aligned}$$

$$\begin{aligned}
p_1 \swarrow o_0 &= p_1 \\
p_1 \swarrow (p_0 \oplus_0 p_0') &= (p_1 \swarrow p_0) \swarrow p_0' \\
o_1 \swarrow p_0 &= o_1 \\
(p_1 \oplus_1 p_1') \swarrow p_0 &= (p_1 \swarrow (p_1' \searrow p_0)) \oplus_1 (p_1' \swarrow p_0)
\end{aligned}$$

Zappa-Szép product of monoids

- A *Zappa-Szép product* (aka *knit product*, *bicrossed product*, *bilateral semidirect product*) of two monoids $(P_0, o_0, \oplus_0)$ and $(P_1, o_1, \oplus_1)$ is a monoid structure $(o, \oplus)$ on $P = P_0 \times P_1$ such that

$$\begin{gathered} o = (o_0, o_1) \\ (p, o_1) \oplus (p', o_1) = (p \oplus_0 p', o_1) \\ (o_0, p) \oplus (o_0, p') = (o_0, p \oplus_1 p') \\ (p, o_1) \oplus (o_0, p') = (p, p') \end{gathered}$$

- Zappa-Szép products of $(P_0, o_0, \oplus_0)$ and $(P_1, o_1, \oplus_1)$ are in a bijective correspondence with matching pairs of actions of $(P_0, o_0, \oplus_0)$ and $(P_1, o_1, \oplus_1)$.
- Given $\oplus$, one constructs $\searrow$ and $\swarrow$ by
    - $(p_1 \searrow p_0, p_1 \swarrow p_0) = (o_0, p_1) \oplus (p_0, o_1)$
- Given $\searrow$ and $\swarrow$, $\oplus$ is defined by
    - $(p_0, p_1) \oplus (p_0', p_1') = (p_0 \oplus_0 (p_1 \swarrow p_0'), (p_1 \searrow p_0') \oplus_1 p_1')$

Two writer monads

- Compatible compositions of writer monads for $(P_0, \mathrm{o}_0, \oplus_0)$ and $(P_1, \mathrm{o}_1, \oplus_1)$ are in a bijection with matching pairs of actions of the two monoids.
- They are isomorphic to writer monads for the corresponding Zappa-Szép products.

Combining popping and pushing

- Take $(P_0, o_0, \oplus_0) = (\mathsf{Nat}, 0, +)$,
  $(P_1, o_1, \oplus_1) = (\mathsf{List}\, E, [], +\!\!+)$ where E is some set.

- $es \searrow n = n \mathbin{\dot{-}} \mathit{length}\, es$,
  $es \swarrow n = \mathit{removelast}\; n\; es$.
- $(n, es) \oplus (n', es')$
  $= (n + (n' \mathbin{\dot{-}} \mathit{length}\, es), (\mathit{removelast}\; n'\; es) +\!\!+ es')$

- Pairs (n, es) represent net effects of sequences of pop, push instructions on a stack: some number of elements is removed from and some new specific elements are added to the stack.

Combining reading, popping, pushing

- How do I now show that
  - $TA = \mathsf{List}\, E \Rightarrow \mathsf{Nat} \times (\mathsf{List}\, E \times A)$

  is a monad?
- This is of the form $T_0 \cdot T_1 \cdot T_2$ where
  - $T_0 A = \mathsf{List}\, E \Rightarrow A$
  - $T_1 A = \mathsf{Nat} \times A$
  - $T_2 A = \mathsf{List}\, E \times A$
- We already know that
  - $T_{01} = T_0 \cdot T_1$
  - $T_{02} = T_0 \cdot T_2$
  - $T_{12} = T_1 \cdot T_2$

  are compatible compositions of monads.
- We want to be sure that $(T_0 \cdot T_1) \cdot T_2$ and $T_0 \cdot (T_1 \cdot T_2)$ are compatible compositions of monads.
- Moreover they'd better be the same monad!

- In terms of distributive laws, this only requires checking the Yang-Baxter equation:

$$
\begin{tikzcd}
 & T_1 \cdot T_2 \cdot T_0 \arrow[r, "T_1 \cdot \theta_{02}"] \arrow[drr, "\theta_{0(12)}"] & T_1 \cdot T_0 \cdot T_2 \arrow[dr, "\theta_{01} \cdot T_2"] & \\
T_2 \cdot T_1 \cdot T_0 \arrow[ur, "\theta_{12} \cdot T_0"] \arrow[urr, "\theta_{(10)2}"] \arrow[dr, "T_2 \cdot \theta_{01}"'] \arrow[drr, "\theta_{0(21)}"] & & & T_0 \cdot T_1 \cdot T_2 \\
 & T_2 \cdot T_0 \cdot T_1 \arrow[r, "\theta_{02} \cdot T_1"'] \arrow[urr, "\theta_{(01)2}"] & T_0 \cdot T_2 \cdot T_1 \arrow[ur, "T_0 \cdot \theta_{12}"'] &
\end{tikzcd}
$$