Containers for Effects and Contexts: Lecture 1

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
 - 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.)

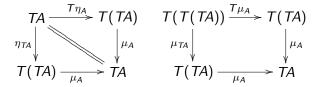


Monads

Monads

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

such that



• This definition says that monads are monoids in the monoidal category ($[C, C], Id_C, \cdot$).

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 $k^*:TA\to TB$ (the *Kleisli extension* operation)

such that

- if $k: A \to TB$, then $k^* \circ \eta_A = k$,
- $\eta_A^{\star} = \operatorname{id}_{TA}$,
- if $k: A \to TB$, $\ell: B \to TC$, then $(\ell^* \circ k)^* = \ell^* \circ k^* : TA \to TC$.
- (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 $k^* =_{\mathrm{df}} TA \xrightarrow{Tk} T(TB) \xrightarrow{\mu_B} TB$.

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

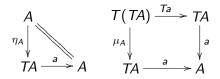
Kleisli category of a monad

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

- From C there is an identity-on-objects inclusion functor J
 to KI(T), defined on maps by
 - if $f: A \to B$, then $Jf = \underset{\text{df}}{\xrightarrow{f}} A \xrightarrow{f} B \xrightarrow{\eta_B} TB = A \xrightarrow{\eta_A} TA \xrightarrow{Tf} TB.$

Monad algebras

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



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

$$TA \xrightarrow{Th} TB$$

$$\downarrow b$$

$$A \xrightarrow{h} B$$

 The algebras of the monad and maps between them form a category EM(T) with an obvious forgetful functor U: EM(T) → C.

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



Kleisli adjunction

- In the opposite direction of J: C → KI(T) there is a functor R: KI(T) → C defined by
 - $RA =_{\mathrm{df}} TA$,
 - if $k: A \to^T B$, then $Rk =_{df} TA \xrightarrow{k^*} TB$.
- R is right adjoint to J.

$$\begin{array}{ccc}
KI(T) & & \stackrel{JA}{\overbrace{A \to^T B}} \\
\downarrow C & & & \overline{A \to TB} \\
\hline
RB & & & & \\
\hline
RB & & & & \\
RB & & & & \\
\end{array}$$

- Importantly, $R \cdot J = T$. Indeed,
 - R(JA) = TA,
 - if $f: A \to B$, then $R(Jf) = (\eta_B \circ f)^* = Tf$.
- 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 there is a functor
 L: C → EM(T) defined by
 - $LA =_{df} (TA, \mu_A)$,
 - if $f: A \to B$, then $Lk =_{\mathrm{df}} Tf: (TA, \mu_A) \to (TB, \mu_B)$.
- L is left adjoint to U.

$$EM(T)$$

$$L \left(\begin{array}{c} A \\ A \end{array} \right) U \qquad \underbrace{(TA, \mu_A) \to (B, b)}_{U(B, b)}$$

- $U \cdot L = T$. 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:
 - $TA =_{df} E + A$ where E is some set (of exceptions)
- The monad structure:
 - $\eta_A x =_{\mathrm{df}} \operatorname{inr} x$,
 - μ_A (inl e) =_{df} inl e, μ_A (inr (inl e)) =_{df} inl e, μ_A (inr (inr x)) =_{df} 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 $o =_{df} ?_E : 0 \to E$ and $\oplus =_{df} \nabla_E : E + E \to E$.)

Reader monads

- The functor:
 - $TA =_{\mathrm{df}} S \Rightarrow A$ where S is a set (of readable states)
- The monad structure:
 - $\eta_A x =_{\mathrm{df}} \lambda s. x$,
 - $\mu_A f =_{\mathrm{df}} \lambda s. f s s.$
- 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:
 - $TA =_{\mathrm{df}} P \times A$ where P is a set (of updates)
- The possible monad structures are:
 - $\eta_A x =_{df} (o, x),$
 - $\mu_A(p,(p',x)) =_{\mathrm{df}} (p \oplus p',x)$ where (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:
 - $TA =_{\mathrm{df}} S \Rightarrow S \times A$ where S is a set (of readable/overwritable states),
 - $\eta_A x =_{\mathrm{df}} \lambda s. (s, x)$
 - $\mu_A f =_{\mathrm{df}} \lambda s$. let (s', g) = f s in g(s', x)

This example works in any monoidal closed category.

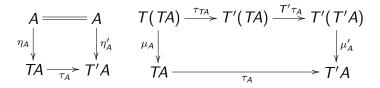
List monad and variations

- The list monad:
 - $TA =_{df} List A$,
 - $\eta_A x =_{\mathrm{df}} [x]$,
 - $\mu_A xss =_{df} concat xs$.
- Some variations:
 - $TA =_{df} \{xs : A^* \mid xs \text{ is square-free}\}$
 - $TA =_{df} \{xs : A^* \mid xs \text{ is duplicate-free}\}$
 - $TA =_{df} 1 + A \times A$
 - $TA =_{df} \mathcal{M}_f A$
 - $TA =_{\mathrm{df}} \mathcal{P}_{\mathrm{f}} A$
 - 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 $\tau: T \to T'$ satisfying



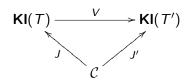
- Monads on $\mathcal C$ and maps between them form a category $\mathbf{Monad}(\mathcal C)$.
- Monad maps are monoid maps in the monoidal category $([\mathcal{C},\mathcal{C}], \mathrm{Id}_{\mathcal{C}}, \cdot)$ and the category of monads is the category of monoids in $([\mathcal{C},\mathcal{C}], \mathrm{Id}_{\mathcal{C}},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

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
 - $\bullet \ \tau_{A} \circ \eta_{A} = \eta'_{A},$
 - if $k: A \to TB$, then $\tau_B \circ k^* = (\tau_B \circ k)^{*\prime} \circ \tau_A$.
- (No explicit naturality condition on τ !)
- Kleisli triples on C and maps between them form a category that is isomorphic to $\mathbf{Monad}(C)$.

Monad maps vs. functors between Kleisli categories

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



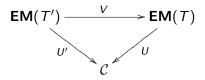
- This is defined by
 - $VA =_{df} A$,
 - if $k: A \to TB$, then $Vk =_{\mathrm{df}} A \xrightarrow{k} TB \xrightarrow{\tau_B} T'B$.

and

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

Monad maps vs. functors between E-M categories

There is a bijection between monad maps τ : T → T'
and functors V : EM(T') → EM(T) such that



(Note the reversed direction.)

- This is defined by
 - $V(A, a) =_{\mathrm{df}} (A, a \circ \tau_A),$
 - if $h: (A, a) \rightarrow (B, b)$, then $Vh =_{\mathrm{df}} h: (A, a \circ \tau_A) \rightarrow (B, b \circ \tau_B)$.

and

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

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, o, ⊕) and (P', o', ⊕') 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, 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, o, \oplus) to the exception monad for E are in a bijection between monoid homomorphisms between (P, 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 $lkp: C \rightarrow S$, $upd: C \times S \rightarrow C$ such that
 - lkp(upd(c,s)) = s,
 - upd(c, lkpc)) = 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

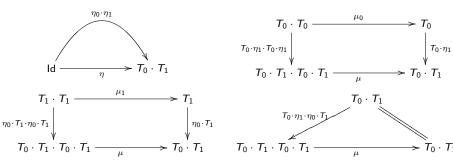
- How many monad maps are there from the nonempty list monad to itself?
- Answer: 6, viz. the identity map, reverse, take the first and last elements, take the last and first elements, take only the first element, take only the last element.
- Why does taking the 2nd element not qualify? Or taking the 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

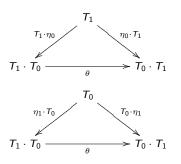
• A compatible composition of two monads (T_0, η_0, μ_0) , (T_1, η_1, μ_1) is a monad structure (η, μ) on $T =_{\mathrm{df}} T_0 \cdot T_1$ satisfying

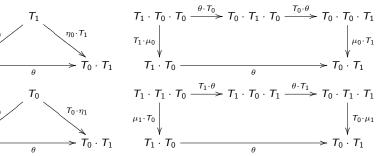


• Conditions 1-3 say just that $T_0 \cdot \eta_1$ and $\eta_0 \cdot T_1$ are monad morphisms between (T_0, η_0, μ_0) resp. (T_1, η_1, μ_1) and (T, η, μ) . Condition 1 fixes that $\eta = \eta_0 \cdot \eta_1$; so the only freedom is about μ .

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





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

$$T_{1}A \stackrel{a_{1}}{\longleftarrow} A \stackrel{a_{0}}{\longrightarrow} T_{0}A$$

$$T_{1}a_{0} \downarrow \qquad \qquad \downarrow T_{0}a_{1}$$

$$T_{1}(T_{0}A) \stackrel{\theta_{A}}{\longrightarrow} T_{0}(T_{1}A)$$

- Such pairs of algebras are in a bijection with (T, η, μ) -algebras.
- Given a_0 , a_1 , one constructs a as

•
$$a =_{df} T_0(T_1A) \xrightarrow{T_0a_1} T_0A \xrightarrow{a_0} A$$
.

• Given a_1 and a_1 are defined by

•
$$a_0 =_{\mathrm{df}} T_0 A \xrightarrow{T_0 \eta_1} T_0(T_1 A) \xrightarrow{a} A$$
,

•
$$a_1 =_{\mathrm{df}} T_1 A \xrightarrow{T_1 \eta_0} T_1(T_0 A) \xrightarrow{\theta_A} T_0(T_1 A) \xrightarrow{a} A.$$



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) .
- $\begin{array}{l} \bullet \ \theta : E + T_0 A \rightarrow T_0 (E + A) \\ \theta_A (\mathsf{inl} \ e) =_{\mathrm{df}} \eta_0 (\mathsf{inl} \ e), \\ \theta_A (\mathsf{inr} \ c) =_{\mathrm{df}} T_0 \mathsf{inr} \end{array}$
- So we have a unique monad structure on $TA =_{\mathrm{df}} T_0(E+A)$ 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, o, \oplus) over any monad (T_0, η_0, μ_0) .
- $\theta: P \times T_0 A \rightarrow T_0(P \times A)$ $\theta_A(p,c) =_{\mathrm{df}} T_0(\lambda x.(p,x))c).$ (θ is nothing but the unique strength of T_0 !)
- So monad structures on $TA =_{\mathrm{df}} T_0(P \times A)$ 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, o, \oplus) on a set S is a map $\downarrow : S \times P \rightarrow S$ satisfying

$$s \downarrow o = s
s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$$

Reader and writer monads

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

$$TA =_{\mathrm{df}} S \Rightarrow P \times A$$
 $\eta x =_{\mathrm{df}} \lambda s. (o, x)$
 $\mu f =_{\mathrm{df}} \lambda s.$ let $(p, g) = f s$
 $(p', x) = g (s \downarrow p)$
 $in (p \oplus p', x)$

—the update monad for S, (P, o, \oplus) , \downarrow .

State logging

- Take *S* to be some set (of states).
- Take $P =_{\mathrm{df}} \mathrm{List}\, S$, $\mathrm{o} =_{\mathrm{df}} []$, $\oplus =_{\mathrm{df}} +++$ (state logs).
- Set

$$s \downarrow [] =_{\mathrm{df}} s$$

 $s \downarrow (s' :: ss) =_{\mathrm{df}} s' \downarrow ss$

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

Reading a stack and popping

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

Matching pairs of monoid actions

A matching pair of actions of two monoids (P₀, o₀, ⊕₀) and (P₁, o₁, ⊕₁) on each other is pair of maps
 ∴ : P₁ × P₀ → P₀ and ✓ : P₁ × P₀ → P₁ such that

$$egin{aligned} \mathsf{o}_1 \searrow p_0 &= p_0 \ (p_1 \oplus_1 p_1') \searrow p_0 &= p_1 \searrow (p_1' \searrow p_0) \ p_1 \searrow \mathsf{o}_0 &= \mathsf{o}_0 \ \end{pmatrix} \ p_1 \searrow (p_0 \oplus_0 p_0') &= (p_1 \searrow p_0) \oplus_0 ((p_1 \swarrow p_0) \searrow p_0') \ p_1 \swarrow (p_0 \oplus_0 p_0') &= (p_1 \swarrow p_0) \swarrow p_0' \ p_1 \swarrow p_0 &= \mathsf{o}_1 \ p_1 \swarrow p_0 &= \mathsf{o}_1 \ p_1 \otimes_1 p_1' \otimes_1 p_0 &= (p_1 \swarrow p_0') \oplus_1 (p_1' \swarrow p_0) \end{pmatrix} \oplus_1 (p_1' \swarrow p_0) \oplus_1 (p_1$$

Zappa-Szép product of monoids

• A Zappa-Szép product (or bicrossed product) of two monoids (P_0, o_0, \oplus_0) and (P_1, o_1, \oplus_1) is a monoid structure (o, \oplus) on $P =_{\mathrm{df}} P_0 \times P_1$ such that

$$egin{aligned} oldsymbol{\circ} &= ig(oldsymbol{\circ}_0, oldsymbol{\circ}_1 ig) \ ig(oldsymbol{\rho}, oldsymbol{\circ}_1 ig) \oplus ig(oldsymbol{o}_0, oldsymbol{\rho}' ig) &= ig(oldsymbol{o}_0, oldsymbol{\rho} oldsymbol{\circ}_1, oldsymbol{o}_1 ig) \ ig(oldsymbol{\rho}, oldsymbol{\circ}_1 ig) \oplus ig(oldsymbol{o}_0, oldsymbol{\rho}' ig) &= ig(oldsymbol{\rho}, oldsymbol{\rho}' ig) \end{aligned}$$

- 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 • $(p_1 \searrow p_0, p_1 \swarrow p_0) =_{df} (o_0, p_1) \oplus (p_0, o_1)$
- ullet Given \searrow and \swarrow , \oplus is defined by
 - $(p_0, p_1) \oplus (p'_0, p'_1) =_{\mathrm{df}} (p_0 \oplus_0 (p_1 \swarrow p'_0), (p_1 \searrow p'_0) \oplus_1 p'_1)$

Two writer monads

- Compatible compositions of writer monads for (P_0, o_0, \oplus_0) and (P_1, 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 $(P_0, o_0, \oplus_0) =_{\mathrm{df}} (\mathsf{Nat}, 0, +)$, $(P_1, o_1, \oplus_1) =_{\mathrm{df}} (\mathsf{List}\, E, [], ++)$ where E is some set.
- es $\searrow n =_{\mathrm{df}} n length$ es, es $\swarrow n =_{\mathrm{df}} removelast n$ es.
- $(n, es) \oplus (n', es')$ = $_{df} (n + (n' - length es'), (removelast n' es)++es')$
- Pairs (n, es) represent net effects of sequences of pop, push instruction on a stack: some number of elements is removed from and some new specific elements are added to the stack.