One Monad to the Tune of Another

real title: Dijkstra Monads For All

Robert Atkey
Strathclyde University, Glasgow

jww Kenji Maillard, Danel Ahman, Guido Martínez,
Cătălin Hriţcu, Exequiel Rivas, and Éric Tanter

Shonan Meeting 146
27th March 2019

Modelling programs:

$$[\![P]\!] : X \longrightarrow Y$$

Modelling programs with side effects:

State

$$[\![P]\!] : X \times S \longrightarrow Y \times S$$

Exceptions

$$[\![P]\!] : X \longrightarrow Y + E$$

Non-determinism

$$[\![P]\!] : X \longrightarrow \mathcal{P}_{\mathit{fin}}(Y)$$

I/O

$$[\![P]\!] : X \longrightarrow \mathrm{IOTree}(Y)$$

Monads as *Notions of Computation*:

(Moggi, 1989, 1991)

$$[\![P]\!] : X \longrightarrow MY$$

Monadic basics:

$$\begin{aligned} \mathit{return} &: A \longrightarrow MA \\ \mathit{bind} &: MA \longrightarrow (A \longrightarrow MA) \longrightarrow MB \end{aligned}$$

Pick your monad:

Nothing :	$MA = A$
State :	$MA = S \longrightarrow (A \times S)$
Exceptions :	$MA = A + E$
Non-determinism :	$MA = \mathcal{P}_{fin}(A)$
I/O :	$MA = \mathrm{IOTree}(A)$

Reasoning about monadic computations

Equational Reasoning

. Monad laws

. Monads generated from equational theories $\rightsquigarrow$ equations for reasoning.

. (Plotkin & Pretnar, 2008)

Evaluation Logic

(Pitts, 1991)

. only allows reasoning about returned values

Indexed Monads

. {Parameterised / Graded / Poly}monads

Special Purpose Logics

. Hoare Type Theory

. Dynamic Logic

. ...

Monads as Notions of Specification

Specification Monads

Many *notions of specification* are also monads.

Specification Monads

Many *notions of specification* are also monads.

Weakest precondition transformer

$$WP(A) = (A \to \mathrm{Prop}) \to \mathrm{Prop}$$

Specification Monads

Many *notions of specification* are also monads.

Weakest precondition transformer

$$WP(A) = (A \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop}$$

Weakest precondition transformer with state

$$WP_S(A) = (A \longrightarrow S \longrightarrow \mathrm{Prop}) \longrightarrow S \longrightarrow \mathrm{Prop}$$

Specification Monads

Many *notions of specification* are also monads.

Weakest precondition transformer

$$WP(A) = (A \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop}$$

Weakest precondition transformer with state

$$WP_S(A) = (A \longrightarrow S \longrightarrow \mathrm{Prop}) \longrightarrow S \longrightarrow \mathrm{Prop}$$

Weakest precondition transformer with exceptions

$$WP_E(A) = (A + E \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop} \cong (A \longrightarrow \mathrm{Prop}) \longrightarrow (E \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop}$$

Specification Monads

Many *notions of specification* are also monads.

Specification Monads

Many *notions of specification* are also monads.

Predicate Monad

$$P(A) = A \longrightarrow \mathrm{Prop}$$

Specification Monads

Many *notions of specification* are also monads.

Predicate Monad

$$P(A) = A \longrightarrow \mathrm{Prop}$$

Pre-/Post-condition Monad

$$PP(A) = \mathrm{Prop} \times (A \longrightarrow \mathrm{Prop})$$

Specification Monads

Many *notions of specification* are also monads.

Predicate Monad

$$P(A) = A \longrightarrow \mathrm{Prop}$$

Pre-/Post-condition Monad

$$PP(A) = \mathrm{Prop} \times (A \longrightarrow \mathrm{Prop})$$

For now, will use the weakest precondition transformer monads for specifications.

Specification Monads W

Useful specification monads are *ordered*:

1. partial order structure $\sqsubseteq$ on WX
2. compatible with the *bind* operation

For WP, must restrict to *monotone* predicate transformers:

$$MonoWP(A) = \{f : (A \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop} \mid \forall q_1, q_2 . q_1 \sqsubseteq q_2 \Rightarrow f\ q_1 \sqsubseteq f\ q_2\}$$

then

$$f \sqsubseteq f' \Leftrightarrow \forall q : A \longrightarrow \mathrm{Prop}.\ f\ q \sqsubseteq f'\ q$$

Connecting the Computational and the Specificational

Connecting the Computational and the Specificational

Monad morphisms:

$$\alpha : M \Rightarrow W$$

Families of functions $\alpha_A : MA \longrightarrow WA$, compatible with *return* and *bind*.

Monad morphisms are “Effect Observations” in (Katsumata, 2014)
(used to generate graded monads)

Connecting the Computational and the Specificational

Monad morphisms:

$$\alpha : M \Rightarrow W$$

Families of functions $\alpha_A : MA \longrightarrow WA$, compatible with *return* and *bind*.

Monad morphisms are “Effect Observations” in (Katsumata, 2014)
(used to generate graded monads)

$$\text{Given} \quad \left.\begin{array}{l} \text{a computation } m : MA \\ \text{a specification } w : WA \end{array}\right\} \qquad m \text{ satisfies } w \quad \text{iff} \quad \alpha_A(m) \sqsubseteq w$$

Example: Non-determinism

The non-determinism monad:

$$ND(A) = \mathcal{P}_{fin}(A)$$

Two morphisms $ND \Rightarrow MonWP$:

Demonic

$$\alpha^{D}(S) = \lambda post.\ \forall a \in S.\ post(a)$$

Angelic

$$\alpha^{A}(S) = \lambda post.\ \exists a \in S.\ post(a)$$

Demonic and Angelic Specifications

For the specification

$$w = (\lambda post.\ \forall x\ y\ z.\ x^2 + y^2 = z^2 \Rightarrow post(x, y, z)) \in MonWP(\mathbb{N} \times \mathbb{N} \times \mathbb{N})$$

$$\begin{aligned}\alpha^D(&\textbf{let } x \leftarrow pick\ [1, 2, 3, 4, 5] \\ &\textbf{let } y \leftarrow pick\ [1, 2, 3, 4, 5] \\ &\textbf{let } z \leftarrow pick\ [1, 2, 3, 4, 5] \\ &\textbf{guard } (x^2 + y^2 = z^2) \\ &\textbf{return } (x, y, z)) \sqsubseteq w\end{aligned}$$

Demonic and Angelic Specifications

For the specification

$$w = (\lambda post.\ \forall x\ y\ z.\ x^2 + y^2 = z^2 \Rightarrow post(x, y, z)) \in \mathit{MonWP}(\mathbb{N} \times \mathbb{N} \times \mathbb{N})$$

$$\begin{aligned}
\alpha^A(&\mathbf{let}\ x \leftarrow \mathit{pick}\ [1, 2, 3, 4, 5] \\
&\mathbf{let}\ y \leftarrow \mathit{pick}\ [1, 2, 3, 4, 5] \\
&\mathbf{let}\ z \leftarrow \mathit{pick}\ [1, 2, 3, 4, 5] \\
&\mathbf{guard}\ (x^2 + y^2 = z^2) \\
&\mathbf{return}\ (x, y, z)) \sqsubseteq w
\end{aligned}$$

Demonic and Angelic Specifications

For the specification

$$w = (\lambda post.\ \forall x\ y\ z.\ x^2 + y^2 = z^2 \Rightarrow post(x, y, z)) \in MonWP(\mathbb{N} \times \mathbb{N} \times \mathbb{N})$$

$$\begin{aligned}\alpha^A(&\mathbf{let}\ x \leftarrow pick\ [1, 2, 3, 4, 5]\\ &\mathbf{let}\ y \leftarrow pick\ [1, 2, 3, 4, 5]\\ &\mathbf{let}\ z \leftarrow pick\ [1, 2, 3, 4, 5]\\ \\ &\mathbf{return}\ (x, y, z)) \sqsubseteq w\end{aligned}$$

Example: State

$$\begin{array}{rclcl}
\alpha_A & : & \mathrm{St}(A) & \longrightarrow & \mathrm{MonWP}_S(A) \\
\alpha_A & : & (S \longrightarrow (S \times A)) & \longrightarrow & (A \longrightarrow S \longrightarrow \mathrm{Prop}) \longrightarrow_{mon} S \longrightarrow \mathrm{Prop}
\end{array}$$

$$\alpha_A \quad m \quad = \quad \lambda post\ s_0 . post(m\ s_0)$$

Spec: $w = (\lambda post\ s_0.\ \forall s.\ s > s_0 \Rightarrow post(s, *)) \in \mathrm{MonWP}_S(1)$

$$\alpha_1(\mathbf{let}\ x \leftarrow get; put(x+1)) \sqsubseteq w$$

Example: Exceptions

Example: Exceptions

1) *"Double-Barrelled" specifications*

$$\begin{aligned}
\alpha_A &: \mathrm{Exc}(A) &\longrightarrow&\; \mathrm{MonWP}_E(A) \\
\alpha_A &: A + E &\longrightarrow&\; (A + E \to \mathrm{Prop}) \to_{mon} \mathrm{Prop}
\end{aligned}$$

$$\alpha_A \quad m \quad = \quad \lambda post.\ post(m)$$

Example: Exceptions

1) *"Double-Barrelled" specifications*

$$
\begin{array}{lclcl}
\alpha_A & : & \mathrm{Exc}(A) & \longrightarrow & \mathrm{MonWP}_E(A) \\
\alpha_A & : & A + E & \longrightarrow & (A + E \longrightarrow \mathrm{Prop}) \longrightarrow_{mon} \mathrm{Prop} \\
 & & & & \\
\alpha_A & & m & = & \lambda post.\ post(m)
\end{array}
$$

2) *"Single-Barrelled" specifications*, for a fixed $Q_{exn} : E \longrightarrow \mathrm{Prop}$

$$
\begin{array}{lclcl}
\alpha_A & : & \mathrm{Exc}(A) & \longrightarrow & \mathrm{MonWP}(A) \\
\alpha_A & : & A + E & \longrightarrow & (A \longrightarrow \mathrm{Prop}) \longrightarrow_{mon} \mathrm{Prop} \\
 & & & & \\
\alpha_A & & m & = & \lambda post.\ \mathbf{case}\ m\ \{\mathrm{inl}\ a \mapsto post(a); \mathrm{inr}\ e \mapsto Q_{exn}(e)\}
\end{array}
$$

Example: I/O

Let $Trace = List(I + O)$,

$$MonWPTrace(A) = (A \to Trace \to \mathrm{Prop}) \to_{mon} Trace \to \mathrm{Prop}$$

$$\begin{array}{lcl} \alpha : \mathrm{IOTree}(A) & \to & MonWPTrace(A) \\ \alpha(return\ a) & = & \lambda post\ t.\ post\ a\ t \\ \alpha(inp\ k) & = & \lambda post\ t.\ \forall i.\ \alpha(k\ i)\ post\ (\mathrm{In}\ i :: t) \\ \alpha(out\ o\ k) & = & \lambda post\ t.\ \alpha(k)\ post\ (\mathrm{Out}\ o :: t) \end{array}$$

Example (assuming $I = O$):

$$\alpha(\mathbf{let}\ x \leftarrow inp; out\ x; out\ x) \sqsubseteq \lambda p\ t.\ \forall x.\ p * (\mathrm{Out}\ x :: \mathrm{Out}\ x :: \mathrm{In}\ x :: t)$$

Mass producing Effect Observations

Effect Observations from Monad Algebras

Monad Algebras

$$h : MR \longrightarrow R$$

are in 1-1 correspondence with monad morphisms

$$\alpha : M \Rightarrow (- \longrightarrow R) \longrightarrow R$$

(Kelly, 1980; Kelly and Power, 1993) ... and extends to the ordered case.

$$\alpha(m) = \lambda k.h(bind_M(m, k))$$

Effect Observations from Monad Algebras

Demonic non-determinism:

$$h : \mathcal{P}_{fin}(\mathrm{Prop}) \longrightarrow \mathrm{Prop}$$
$$h(S) = \bigwedge S$$

gives

$$\alpha^{D}(S) = \lambda post.\ \forall x \in S.\ post(x)$$

Angelic non-determinism:

$$h : \mathcal{P}_{fin}(\mathrm{Prop}) \longrightarrow \mathrm{Prop}$$
$$h(S) = \bigvee S$$

gives

$$\alpha^{A}(S) = \lambda post.\ \exists x \in S.\ post(x)$$

Effect Observations from Monad Algebras

State:

$$h : \mathrm{St}(S \longrightarrow \mathrm{Prop}) \longrightarrow S \longrightarrow \mathrm{Prop}$$
$$h(t) = \lambda s.\ \mathrm{let}\ (s', p) = t\ s\ \mathrm{in}\ p\ s'$$

gives

$$\alpha(t) = \lambda post.\lambda s.\mathrm{let}\ (s', a) = t\ s\ \mathrm{in}\ post\ s'$$

Effect Observations from Monad Algebras

Single-barrelled Exceptions:

$$
\begin{array}{l}
h : \mathrm{Exc}(\mathrm{Prop}) \longrightarrow \mathrm{Prop} \\
h\,(\mathrm{inl}\ p) \quad = \quad p \\
h\,(\mathrm{inr}\ e) \quad = \quad Q_{exn}\ e
\end{array}
$$

gives

$$\alpha_A\, m = \lambda post.\ \mathbf{case}\ m\ \{\mathrm{inl}\ a \mapsto post(a); \mathrm{inr}\ e \mapsto Q_{exn}(e)\}$$

Example: Free Monad

Assume $\Sigma = \{op_1 : I_1 \rightsquigarrow O_1, \cdots, op_n : I_n \rightsquigarrow O_n\}$.

$$T_\Sigma A = \mu X. A + \coprod_{op:I \rightsquigarrow O \in \Sigma} I \times (O \to X)$$

Example: Free Monad

Assume $\Sigma = \{op_1 : I_1 \rightsquigarrow O_1, \cdots, op_n : I_n \rightsquigarrow O_n\}$.

$$T_\Sigma A = \mu X.A + \coprod_{op:I \rightsquigarrow O \in \Sigma} I \times (O \to X)$$

Operation specifications, for all op: $\quad pre_{op} : I \to \text{Prop}$ and $post_{op} : I \to O \to \text{Prop}$

$$\begin{aligned}
&\alpha : T_\Sigma \text{Prop} \to \text{Prop} \\
&\alpha(ret\ \phi) = \phi \\
&\alpha(op\ i\ k) = pre_{op}\ i \wedge \forall o.\ post_{op}\ i\ o \to k\ o
\end{aligned}$$

Monad Transformers

Assume that we have a monad transformer:

$$\mathcal{T} : \text{Mon} \longrightarrow \text{Mon}$$

. functor from monads to monads
. equipped with lift : $M \Rightarrow \mathcal{T} M$
. preserving order

Example: $\mathcal{T}(M) = A \mapsto S \longrightarrow M(S \times A)$

Given a suitable $\mathcal{T}$,

$$\alpha = \mathcal{T}(\mathit{return}) : \mathcal{T}(\mathit{Id}) \longrightarrow \mathcal{T}(\text{MonCont})$$

DM4Free (POPL'17) presented this idea in a syntactic way.

Monad Transformers

$$\mathcal{T}(M) = A \mapsto M(A + E)$$

$$\begin{aligned} \text{Then:} \quad \mathcal{T}(\text{Id}) &= \text{Exn} \\ \mathcal{T}(\text{MonCont}) &= A \mapsto (A + E \rightarrow \text{Prop}) \rightarrow \text{Prop} \\ \\ \alpha(m) &= \lambda post.\ post(m) \end{aligned}$$

Also works for State, State(Exn), Exn(State), ...

I/O

$$\mathcal{T}(M) = A \mapsto \mu X.M(A + (O \times X) + (I \longrightarrow X))$$

$$\mathcal{T}(\mathrm{MonProp}) = A \mapsto \mu X.(A + (O \times X) + (I \longrightarrow X) \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop}$$

but this doesn't exist in Set; and would make Coq and F# inconsistent...

Coupling Computations with Specifications

Coupling Computations with Specifications

The specification $\alpha(m) \sqsubseteq w$ could be tricky to check.

Coupling Computations with Specifications

The specification $\alpha(m) \sqsubseteq w$ could be tricky to check.

Restrict to computations m that satisfy a specification w:

$$
\begin{aligned}
&D : (A : \mathrm{Set}) \longrightarrow WA \longrightarrow \mathrm{Set} \\
&D\,A\,w = \{m : MA \mid \alpha(m) \sqsubseteq w\}
\end{aligned}
$$

Coupling Computations with Specifications

The specification $\alpha(m) \sqsubseteq w$ could be tricky to check.

Restrict to computations m that satisfy a specification w:

$$\begin{aligned} &D : (A : \mathrm{Set}) \longrightarrow WA \longrightarrow \mathrm{Set} \\ &D\,A\,w = \{m : MA \mid \alpha(m) \sqsubseteq w\} \end{aligned}$$

Define:

$$\begin{aligned} &\mathit{return} : (x : A) \longrightarrow D\,A\,(\mathit{return}_W\ x) \\ &\mathit{return}\,x = \mathit{return}_M\ a \end{aligned}$$

$$\begin{aligned} &\mathit{bind} : D\,A\,w_1 \longrightarrow ((x : A) \longrightarrow D\,B\,(w_2\ x)) \longrightarrow D\,B\,(\mathit{bind}_W\ w_1\ w_2) \\ &\mathit{bind}\ m_1\ m_2 = \mathit{bind}_M\ m_1\ m_2 \end{aligned}$$

$$\begin{aligned} &\mathit{weaken} : (w_1 \sqsubseteq w_2) \longrightarrow D\,A\,w_1 \longrightarrow D\,A\,w_2 \\ &\mathit{weaken}\,m = m \end{aligned}$$

Dijkstra Monads *over a (specification) monad W*:

$$
\begin{aligned}
&D : (A : \mathrm{Set}) \to WA \to \mathrm{Set} \\
&\mathit{return} : (x : A) \to D\,A\,(\mathit{return}_W\ x) \\
&\mathit{bind} : D\,A\,w_1 \to ((x : A) \to D\,B\,(w_2\ x)) \to D\,B\,(\mathit{bind}_W\ w_1\ w_2) \\
&\mathit{weaken} : (w_1 \sqsubseteq w_2) \to D\,A\,w_1 \to D\,A\,w_2
\end{aligned}
$$

with some laws.

Effect Observations and Dijkstra Monads

$$\mathrm{DMon}(X) \simeq \mathrm{Mon}/W$$

an equivalence of categories.

Algebraic Effects and Handlers

Algebraic Effects

If $op^M : I \times (O \longrightarrow MA) \longrightarrow MA$ is *algebraic*

and $\alpha : M \Rightarrow W$ is an effect observation,

then

$$op^W\,(i, w) = \mu^W(\alpha(op^M\,(i, \lambda o.\ return^M\,(w\,o))))$$

is algebraic, and serves to be the specification for the operation op^M:

$$op^D : (i : I) \longrightarrow (c : (o : O) \longrightarrow D\,A\,(w\,o)) \longrightarrow D\,A\,(op^W\,(i, w))$$

Handlers, attempt 1

$$\begin{aligned}\text{handle} : \; & T_\Sigma A \longrightarrow \\ & (I \times (O \longrightarrow MB) \longrightarrow MB)_{op:I \rightsquigarrow O} \longrightarrow \\ & (A \longrightarrow MB) \longrightarrow \\ & MB\end{aligned}$$

Handlers, attempt 1

$$
\begin{aligned}
\text{handle} : \; & T_\Sigma A \to \\
& (I \times (O \to MB) \to MB)_{op:I \rightsquigarrow O} \to \\
& (A \to MB) \to \\
& MB
\end{aligned}
$$

$$
\begin{aligned}
\text{handle}^D : \; & D_\Sigma \; A \; w_1 \; \to \\
& (H^{W'}_{op} : I \times (O \to W'B) \to W'B)_{op:I \rightsquigarrow O} \to \\
& ((i : I) \to ((o : O) \to D' \, B \, (w \; o)) \to D' \, B \, (h^{W'}_{op} \, (i, w)))_{op:I \rightsquigarrow O} \to \\
& ((a : A) \to D' \, B \, (w_2 \; a)) \to \\
& D' \, B \, (\text{handle} \; w_1 \; (h_{H^{W'}})_* \; w_2)
\end{aligned}
$$

where $h_* : WW'B \to W'B$ whenever $h : TW'B \to W'B$.

not automatic; needs to be established for each $\alpha : M \Rightarrow W$ and W'.

Handlers: lifting algebras

For exceptions, $\alpha : \mathrm{Exn} \Rightarrow \mathrm{ExnWP} = ((- + E \longrightarrow \mathrm{Prop}) \longrightarrow \mathrm{Prop})$

Possible to take $\quad h : \mathrm{Exn}(\mathrm{ExnWP}(B)) \longrightarrow \mathrm{ExnWP}(B)$

to $\quad h_* : \mathrm{ExnWP}(\mathrm{ExnWP}(B)) \longrightarrow \mathrm{ExnWP}(B)$

Handlers: lifting algebras

For exceptions, $\alpha : \mathrm{Exn} \Rightarrow \mathrm{ExnWP} = ((- + E \to \mathrm{Prop}) \to \mathrm{Prop})$

Possible to take $\quad h : \mathrm{Exn}(\mathrm{ExnWP}(B)) \to \mathrm{ExnWP}(B)$

to $\quad h_* : \mathrm{ExnWP}(\mathrm{ExnWP}(B)) \to \mathrm{ExnWP}(B)$

For I/O, $\alpha : \mathrm{IO} \Rightarrow \mathrm{MonWPTrace}$, not possible to do the lifting.

The specification monad

$$\mathcal{T}(\mathrm{MonProp}) = A \mapsto \mu X.(A + (O \times X) + (I \to X) \to \mathrm{Prop}) \to \mathrm{Prop}$$

"works", but doesn't exist in categories/theories of interest.

Handlers, attempt 2

Problem seems to be:

- trying to get the “most general” specification for the handled computation
- then try to instantiate that specification with the spec of the handler
- but we get circularity between the handler behaviour and the handled’s behaviour

A possible solution

- Assume some pre_{op} and $post_{op}$ specification for the operations
- Handler of an operation $op(i, k)$:
  - Assumes $pre_{op}(i)$
  - Must establish $post_{op}(i, o)$ before invoking $k\ o$

Sort of works, but only for handling into a Disjktra monad (can’t write state handler).

Conclusions

- Monads as notions of Specification
- Effect observations = monad morphisms
- Packaged up as Dijkstra Monads

- Algebraic *effects* work well
- Handlers are a mess