Update monads

Danel Ahman, U. of Edinburgh

Tarmo Uustalu, Inst. of Cybernetics, Tallinn

EWSCS, 5 March 2014

Background: Effectful functional programming

- Pure functional programs

$$\frac{f : X \longrightarrow Y \qquad g : Y \longrightarrow Z}{\lambda x.\, g\,(f\,x) : X \longrightarrow Z}$$

- But sometimes the functions do side-effects

$$\frac{f : X \longrightarrow 1 + Y \qquad g : Y \longrightarrow 1 + Z}{\lambda x.\, \mathrm{case}\,(f\,x)\,\mathrm{of}\left\{\mathrm{inl}(\_) \implies \mathrm{inl}\,() \mid \mathrm{inr}(y) \implies g\,y\right\} : X \longrightarrow 1 + Z}$$

- Historically monads have provided such general composition
  - $T : \mathbf{Set} \to \mathbf{Set}$
  - $\eta : \forall\{X\}.\, X \to TX$
  - $(-)^* : \forall\{X, Y\}.\, (X \longrightarrow TY) \longrightarrow (TX \longrightarrow TY)$
- Nowadays we often use algebraic presentations

(more on this later)

Background: Effectful functional programming

- Pure functional programs

$$\frac{f : X \longrightarrow Y \qquad g : Y \longrightarrow Z}{\lambda x.\, g\,(f\,x) : X \longrightarrow Z}$$

- But sometimes the functions do side-effects

$$\frac{f : X \longrightarrow 1 + Y \qquad g : Y \longrightarrow 1 + Z}{\lambda x.\, \mathsf{case}\,(f\,x)\,\mathsf{of}\left\{\mathsf{inl}(\_) \implies \mathsf{inl}\,() \mid \mathsf{inr}(y) \implies g\,y\right\} : X \longrightarrow 1 + Z}$$

- Historically monads have provided such general composition
  - $T : \mathbf{Set} \to \mathbf{Set}$
  - $\eta : \forall\{X\}.\, X \to TX$
  - $(-)^* : \forall\{X, Y\}.\, (X \longrightarrow TY) \longrightarrow (TX \longrightarrow TY)$
- Nowadays we often use algebraic presentations

(more on this later)

Background: Effectful functional programming

- Pure functional programs

$$\frac{f : X \longrightarrow Y \qquad g : Y \longrightarrow Z}{\lambda x.\, g\,(f\,x) : X \longrightarrow Z}$$

- But sometimes the functions do side-effects

$$\frac{f : X \longrightarrow (S \to S \times Y) \qquad g : Y \longrightarrow (S \to S \times Z)}{\lambda x.\, \lambda s.\, g\left(\mathsf{snd}\,(f\,x\,s)\right)\left(\mathsf{fst}\,(f\,x\,s)\right) : X \longrightarrow (S \to S \times Z)}$$

- Historically monads have provided such general composition
  - $T : \mathbf{Set} \to \mathbf{Set}$
  - $\eta : \forall\{X\}.\, X \to TX$
  - $(-)^* : \forall\{X, Y\}.\, (X \longrightarrow TY) \longrightarrow (TX \longrightarrow TY)$
- Nowadays we often use algebraic presentations

(more on this later)

Background: Effectful functional programming

- Pure functional programs

$$\frac{f : X \longrightarrow Y \qquad g : Y \longrightarrow Z}{\lambda x.\, g\,(f\,x) : X \longrightarrow Z}$$

- But sometimes the functions do side-effects

$$\frac{f : X \longrightarrow (S \to S \times Y) \qquad g : Y \longrightarrow (S \to S \times Z)}{\lambda x.\, \lambda s.\, g\left(\mathsf{snd}\,(f\,x\,s)\right)\left(\mathsf{fst}\,(f\,x\,s)\right) : X \longrightarrow (S \to S \times Z)}$$

- Historically monads have provided such an interface:
  - $T : \mathbf{Set} \to \mathbf{Set}$
  - $\eta : \forall\{X\}.\, X \to TX$
  - $(-)^* : \forall\{X, Y\}.\, (X \longrightarrow TY) \longrightarrow (TX \longrightarrow TY)$

- Nowadays we often use algebraic presentations

(more on this later)

Background: Three famous monads

State monad

S – a set

$$T_s X = S \to S \times X$$

$$\mathsf{lkp} : (S \to \mathcal{A}) \to \mathcal{A}$$

$$\mathsf{upd} : S \times \mathcal{A} \to \mathcal{A}$$

Reader monad

S – a set

$$T_r X = S \to X$$

$$\mathsf{lkp} : (S \to \mathcal{A}) \to \mathcal{A}$$

Writer monad

$(P, \mathrm{o}, \oplus)$ – a monoid

$$T_w X = P \times X$$

$$\mathsf{upd} : P \times \mathcal{A} \to \mathcal{A}$$

S – states

P – updates (alt. "programs")

$\mathcal{A}$ – carrier of alg. for $T_{\{s,r,w\}}$

+ some equations

This talk: don't just overwrite. update!

Update monad

S – a set

$(P, o, \oplus)$ – a monoid

$\downarrow$ – an action

$$\mathbf{T\,X = S \to P \times X}$$

Reader monad

S – a set

$$T_r\,X = S \to X$$

lkp : $(S \to \mathcal{A}) \to \mathcal{A}$

Writer monad

$(P, o, \oplus)$ – a monoid

$$T_w\,X = P \times X$$

upd : $P \times \mathcal{A} \to \mathcal{A}$

S – states

P – updates (alt. "programs")

$\mathcal{A}$ – carrier of alg. for $T_{\{s,r,w\}}$

$+$ some equations

Monoids, monoid actions

- A monoid on a set P is given by

$$\mathsf{o} : P,$$
$$\oplus : P \to P \to P,$$

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

- An action of a monoid $(P, \mathsf{o}, \oplus)$ on a set S is given by

$$\downarrow : S \to P \to S$$

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

Update monads

A set S, monoid $(P, \mathrm{o}, \oplus)$ and action $\downarrow$ give an update monad:

$$T\,X = S \to (P \times X)$$

$$\eta : \forall\{X\}.\, X \to (S \to P \times X)$$

$$\eta\, x = \lambda s.\, (\mathrm{o}, x)$$

$$\begin{aligned}(-)^* : \forall\{X, Y\}.\, &(X \to (S \to P \times Y)) \\ &\to (S \to P \times X) \to (S \to P \times Y)\end{aligned}$$

$$\begin{aligned}f^*\, g = \lambda s.\, \mathrm{let}\ \ &(p, x) = g\, s; \\ &(p', y) = f\, x\, (s \downarrow p) \\ \mathrm{in}\ \ &(p \oplus p', y)\end{aligned}$$

Reader and writer monads as instances

- Recall update monads:
  $$T\,X = S \to P \times X$$

- Reader monads: $T_r\,X = S \to X$
  update monads with $(P, \mathrm{o}, \oplus)$ and $\downarrow$ trivial

- Writer monads: $T_w\,X = P \times X$
  update monads with S and $\downarrow$ trivial

State monads as canonically related

- State monads:

  $$T_s X = S \to (S \times X)$$

  embed into and project from update monads

$$T_s X = S \to S \times X \qquad T X = S \to (1 + S) \times X$$

for P the free monoid on the overwrite semigroup $(S, \bullet)$

defined by $s \bullet s' = s'$

Update monad example: logging state

- S – a set *(set of states)*
- $P = S^*$ *(log of states)*
- $\mathrm{o} = []$
- $ss \oplus ss' = ss +\!\!+ ss'$
- $s \downarrow ss = \mathrm{last}(s : ss)$
- $T\,X = S \to (S^* \times X)$

Update monad example: writing into a buffer

- $S = E^* \times \mathrm{Nat}$ *(current buffer content and free space)*
- $P = E^*$ *(new values to write)*
- $\circ = []$
- $p \oplus p' = p +\!\!+\, p'$
- $(s, n) \downarrow p = (s +\!\!+\, (p|n), n - \mathit{length}(p|n))$

($p|n$ defined as p truncated to length n)

- $T\,X = (E^* \times \mathrm{Nat}) \to (E^* \times X)$

Algebras of update monads (cf. algebraic effects)

An algebra of an update monad is a set $\mathcal{A}$ with an operation

$$\mathsf{act} : (S \to P \times \mathcal{A}) \to \mathcal{A}$$

$$a = \mathsf{act}\,(\lambda s.\,(\mathsf{o}, a))$$

$$\begin{aligned}&\mathsf{act}\,(\lambda s.\,(p, \mathsf{act}\,(\lambda s'.\,(p', a))))\\ &\qquad = \mathsf{act}\,(\lambda s.\,(p \oplus p'[s \downarrow p/s'], a[s \downarrow p/s']))\end{aligned}$$

or, equivalently a pair of operations

$$\begin{aligned}\mathsf{lkp} &: (S \to \mathcal{A}) \to \mathcal{A}\\ \mathsf{upd} &: P \times \mathcal{A} \to \mathcal{A}\end{aligned}$$

$$a = \mathsf{lkp}\,(\lambda s.\,\mathsf{upd}(\mathsf{o}, a))$$

$$\mathsf{upd}\,(p, \mathsf{upd}\,(p', a)) = \mathsf{upd}\,(p \oplus p', a)$$

$$\mathsf{lkp}\,(\lambda s.\,\mathsf{upd}\,(p, \mathsf{lkp}\,(\lambda s'.\,a))) = \mathsf{lkp}\,(\lambda s.\,\mathsf{upd}\,(p, a[s \downarrow p/s']))$$

Update monads as compatible compositions

The update monad for S, $(P, \mathrm{o}, \oplus)$, $\downarrow$ is the compatible composition of the

reader monads	and	writer monads
$T_r X = S \to X$		$T_w X = P \times X$

for the distributive law

$$\theta : \forall \{X\}. \, P \times (S \to X) \to (S \to P \times X)$$
$$\theta \, (p, f) = \lambda s. \, (p, f \, (s \downarrow p))$$

Thm. There is a bijection between update monads and distributive laws between reader and write monads.

Update monad algebras as compat. compositions

An algebra of the update monad for S, $(P, \mathrm{o}, \oplus)$, $\downarrow$ is a set $\mathcal{A}$ carrying both the

reader monad algebra

$$\mathsf{lkp} : (S \to \mathcal{A}) \to \mathcal{A}$$

$$\mathsf{lkp}\,(\lambda s.\, a) = a$$

$$\mathsf{lkp}\,(\lambda s.\, \mathsf{lkp}\,(\lambda s'.\, a)) = \mathsf{lkp}\,(\lambda s.\, a[s/s'])$$

writer monad algebras

$$\mathsf{upd} : P \times \mathcal{A} \to \mathcal{A}$$

$$\mathsf{upd}\,(\mathrm{o}, a) = a$$

$$\mathsf{upd}\,(p, \mathsf{upd}\,(p', a)) = \mathsf{upd}\,(p \oplus p', a)$$

satisfying an additional compatibility condition

$$\mathsf{upd}\,(p, \mathsf{lkp}\,(\lambda s'.\, a)) = \mathsf{lkp}\,(\lambda s.\, \mathsf{upd}\,(p, a[s \downarrow p/s']))$$

Buffers and truncation revisited

- $S = E^* \times \mathsf{Nat}$ (*current buffer content and free space*)
- $P = E^*$ (*new values to write*)
- $\mathsf{o} = []$
- $p \oplus p' = p +\!\!+\, p'$
- $(s, n) \downarrow p = (s +\!\!+\, (p|n), n - \mathit{length}(p|n))$

  ($p|n$ *defined as* p *truncated to length* n)

- $T\,X = (E^* \times \mathsf{Nat}) \rightarrow (E^* \times X)$
- How to avoid truncation?

A finer dependently-typed version

- Rather than

$$S \text{ -- a set}$$
$$(P, \mathrm{o}, \oplus) \text{ -- a monoid}$$
$$\downarrow \text{ -- an action}$$

$$T\,X = S \to P \times X$$

consider a *directed container* $(S, P, \downarrow, \mathrm{o}, \oplus)$

$$P \text{ a } S\text{-indexed family,}$$

$$\downarrow : \Pi s : S.\, P\, s \to S$$
$$\mathrm{o} : \Pi\{s : S\}.\, P\, s$$
$$\oplus : \Pi\{s : S\}.\, \Pi p : P\, s.\, P\,(s \downarrow p) \to P\, s$$

$$\mathbf{T\,X = \Pi s : S.\, P\, s \times X}$$

S – states

$P\, s$ – updates *enabled* (or *safe*) in state s

Monads from directed containers

The def. of monad is the same (but with dependent typing):

$$T\,X = \Pi s : S.\, P\,s \times X$$

$$\eta : \forall\{X\}.\, X \to \Pi s : S.\, P\,s \times X$$
$$\eta\,x = \lambda s.\, (o, x)$$

$$(-)^* : \forall\{X, Y\}.\, (X \to \Pi s : S.\, P\,s \times Y) \to (\Pi s : S.\, P\,s \times X) \to (\Pi s : S.\, P\,s \times Y)$$

$$(f)^*(g) = \lambda s.\,\mathsf{let}\ \ (p, x) = g\,s;\ (p', y) = f\,x\,(s \downarrow p)\ \mathsf{in}\ (p \oplus p', y)$$

Formally, it is the co-interpretation of directed containers

$$\langle\!\langle - \rangle\!\rangle^{\mathrm{dc}} : \mathbf{DCont}^{\mathrm{op}} \longrightarrow \mathbf{Monads}(\mathbf{Set})$$

Example: writing into a buffer (a finer version)

- $S = E^* \times \mathsf{Nat}$ (*current buffer content* and *free space*)
- $P\ (s, n) = E^{\leq n}$ (*new values to write*)
- $\mathsf{o} = []$
- $p \oplus p' = p +\!\!+\ p'$
- $(s, n) \downarrow p = (s +\!\!+\ p, n - \mathit{length}(p))$
  - no additional truncation needed!
- $T\ X = \Pi(s, n) : E^* \times \mathsf{Nat}.\ E^{\leq n} \times X$

Conclusion

- Update monads $T\,X = S \to (P \times X)$ are a natural combination of the reader and writer monads
  - from a programming perspective
  - from a monadic perspective
  - from an algebraic perspective

- They are also a special case of a more general dependently-typed version
  - the co-interpretation of directed containers

Connection to Kammar-Plotkin generalization

For a set S, a monoid $(P, \mathrm{o}, \oplus)$, an action $\downarrow$,
Kammar and Plotkin defined a generalized state monad as:

$$T_{KP}\,X = \Pi s : S.(s \downarrow P) \times X$$

$T_{KP}\,X$ is the middle monad in the epi-mono factorization

$$\begin{array}{ccc} T\,X = S \to (P \times X) & \xrightarrow{\tau} & T_s\,X = S \to (S \times X) \\ & \searrow \qquad \nearrow & \\ & T_{KP}\,X = \Pi s : S.(s \downarrow P) \times X & \end{array}$$

of the mon. morphism $\tau = \lambda f.\,\lambda s.\,\mathrm{let}(p, x) = f\ s\ \mathrm{in}\ (s \downarrow p, x)$