Take Action for Your State

Effective Conservative Restrictions

Ohad Kammar
<ohad.kammar@ed.ac.uk>
Gordon Plotkin

SPLS & FitA
November 24, 2010

lfcs | Laboratory for Foundations of Computer Science

THE UNIVERSITY of EDINBURGH informatics

A Tale of Three Monads...

State (S a set):

$$A \mapsto S \to (S \times A) \qquad T \mapsto S \to T(S \times -)$$

Reader (S a set):

$$A \mapsto S \to A \qquad T \mapsto S \to T(-)$$

Writer (M a monoid):

$$A \mapsto M \times A \qquad T \mapsto T(M \times -)$$

Some underlying common structure?

Investigation

- View through Algebraic Theory of Effects.
- Generalise using monoid actions.
- Borrow ideas from our work on type and effect systems.
- Describe underlying structure.

Interface vs. Implementation

Signature $\langle \Sigma, : \rangle$:

Set Σ of *function symbols* and an *arity* function $(:) : \Sigma \to \mathbb{N}$.

Theory/Presentation $\langle \Sigma, E \rangle$:

Signature Σ with a set of *equations* $E \subseteq \Sigma\text{-Terms} \times \Sigma\text{-Terms}$.

Monoids:

Signature:

$$(\cdot) : 2$$
$$e : 0$$

Equations:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$
$$x \cdot e = e \cdot x = x$$

Some Universal Algebra

Lawvere theory:

Essentially the quotient: Σ-Terms$/E$.

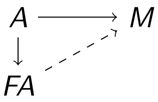
Model $\langle M, [\![-]\!]\rangle$:

Set M and an interpretation of the terms as M-operations.

Free model FA:

Universal property:

All $A \to M$ extend uniquely to $FA \to M$.



Amounts to the quotient Σ-Terms$(A)/E$.

Some Category Theory

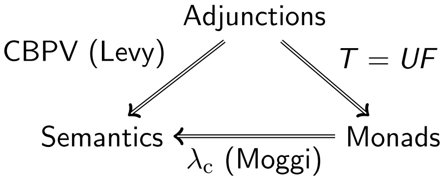
Forgetful Functor:

$$U : \langle M, [\![-]\!] \rangle \mapsto M$$

Free Model Adjunction:

$$F \dashv U$$

Semantics



Summary

Algebraic Theory of Effects

operations, presentations

$\Downarrow$

Lawvere theories

$\Downarrow$

adjunctions

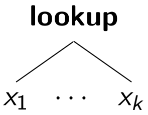
$\Downarrow$

monads

Global State Operations

For a set of states S, $|S| = k > 0$:

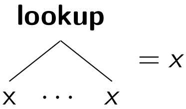
$\textbf{lookup} : |S|$ $\qquad$ $\textbf{upd}_s : 1$



$\textbf{lookup}(\lambda s. x_s)$ $\qquad$ $\textbf{upd}_s\, x$

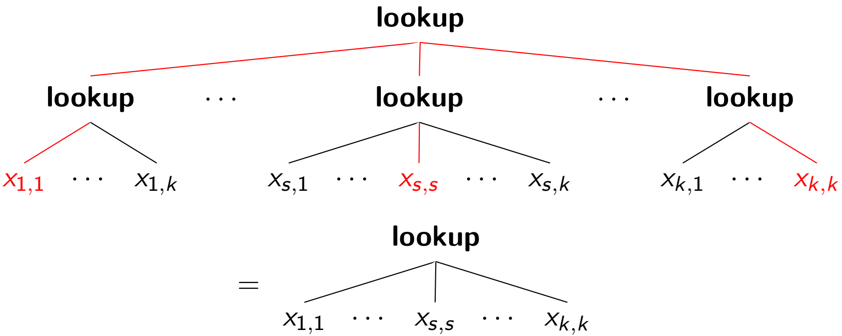
Global State Presentation

$\textbf{lookup}\,(\lambda s.x) = x$



Global State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$



Global State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \quad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

$$\textbf{upd}_s\,(\textbf{upd}_t\,x) = \textbf{upd}_t\,x$$

Laboratory for Foundations of Computer Science

THE UNIVERSITY of EDINBURGH informatics

Global State Presentation

$$\mathbf{lookup}\,(\lambda s.x) = x \quad \mathbf{lookup}\,(\lambda s.\,\mathbf{lookup}\,(\lambda t.x_{s,t})) = \mathbf{lookup}\,(\lambda r.x_{r,r})$$

$$\mathbf{upd}_s\,(\mathbf{upd}_t\,x) = \mathbf{upd}_t\,x \qquad \mathbf{upd}_s\,\mathbf{lookup}\,(\lambda s.x_s) = \mathbf{upd}_s\,x_s$$

Laboratory for Foundations of Computer Science

THE UNIVERSITY of EDINBURGH informatics

Global State Presentation

$$\mathbf{lookup}\,(\lambda s.x) = x \qquad \mathbf{lookup}\,(\lambda s.\,\mathbf{lookup}\,(\lambda t.x_{s,t})) = \mathbf{lookup}\,(\lambda r.x_{r,r})$$

$$\mathbf{upd}_s\,(\mathbf{upd}_t\,x) = \mathbf{upd}_t\,x \qquad \mathbf{upd}_s\,\mathbf{lookup}\,(\lambda s.x_s) = \mathbf{upd}_s\,x_s$$

$$\mathbf{lookup}\,(\lambda s.\,\mathbf{upd}_s\,x_s) = \mathbf{lookup}\,(\lambda s.x_s)$$

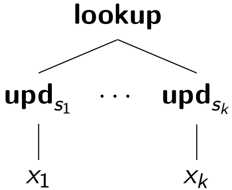
Global State Free Model Monad

$$\mathbf{lookup}\,(\lambda s.x) = x \qquad \mathbf{lookup}\,(\lambda s.\,\mathbf{lookup}\,(\lambda t.x_{s,t})) = \mathbf{lookup}\,(\lambda r.x_{r,r})$$

$$\mathbf{upd}_s\,(\mathbf{upd}_t\,x) = \mathbf{upd}_t\,x \qquad \mathbf{upd}_s\,\mathbf{lookup}\,(\lambda s.x_s) = \mathbf{upd}_s\,x_s$$

$$\mathbf{lookup}\,(\lambda s.\,\mathbf{upd}_s\,x_s) = \mathbf{lookup}\,(\lambda s.x_s)$$

Normal form:



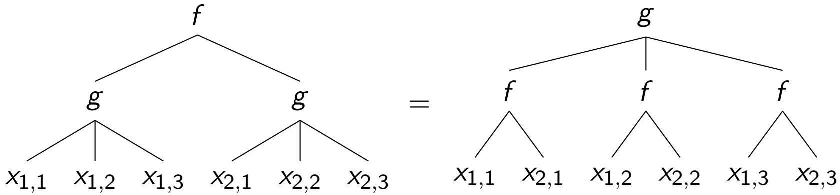
Monad:

$$A \mapsto S \to S \times A$$

Combinations

Tensor

$$\langle \Sigma_1, E_1 \rangle \otimes \langle \Sigma_2, E_2 \rangle = \langle \Sigma_1 + \Sigma_2, \mathrm{Th}(E_1 \cup E_2 \cup E_{\Sigma_1 \otimes \Sigma_2}) \rangle$$



Tensor with {State, Reader, Writer} theory
$\cong$ {State, Reader, Writer} monad transformer.

~~Group~~ Action
Monoid

Right Monoid Action $\langle S, M, \cdot \rangle$:

Set S, monoid M and a function

$$(\cdot) : S \times M \to S$$

compatible with the monoid operation:

$$s \cdot e = s \qquad (s \cdot m) \cdot n = s \cdot (mn)$$

Idea

Generalised state $\leftrightarrow$ Monoid actions

Generalised State Operations

For a monoid action $\langle M, S, \cdot \rangle$, $|S| = k > 0$:

$$\mathbf{lookup} : |S| \qquad \mathbf{act}_m : 1$$

Generalised State Presentation

lookup $(\lambda s.x) = x$

Generalised State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \quad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

Generalised State Presentation

$$\mathbf{lookup}\,(\lambda s.x) = x \quad \mathbf{lookup}\,(\lambda s.\,\mathbf{lookup}\,(\lambda t.x_{s,t})) = \mathbf{lookup}\,(\lambda r.x_{r,r})$$

$$\mathbf{act}_m\,(\mathbf{act}_n\,x) = \mathbf{act}_{mn}\,x$$

Ohad Kammar <ohad.kammar@ed.ac.uk> Gordon Plotkin

Generalised State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \quad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

$$\textbf{act}_m\,(\textbf{act}_n\,x) = \textbf{act}_{mn}\,x$$

$$\textbf{act}_e\,x = x$$

Generalised State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

$$\textbf{act}_m\,(\textbf{act}_n\,x) = \textbf{act}_{mn}\,x \qquad \textbf{act}_m\,\textbf{lookup}\,(\lambda s.x_s) = \textbf{lookup}\,(\lambda r.\,\textbf{act}_m\,x_{r\cdot m})$$

$$\textbf{act}_e\,x = x$$

Generalised State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

$$\textbf{act}_m\,(\textbf{act}_n\,x) = \textbf{act}_{mn}\,x \qquad \textbf{act}_m\,\textbf{lookup}\,(\lambda s.x_s) = \textbf{lookup}\,(\lambda r.\,\textbf{act}_m\,x_{r\cdot m})$$

$$\textbf{act}_e\,x = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{act}_{m_s}\,x_s) = \textbf{lookup}\,(\lambda s.\,\textbf{act}_{n_s}\,x_s)$$

provided for all s, $s \cdot m_s = s \cdot n_s$

Generalised State Presentation

$$\textbf{lookup}\,(\lambda s.x) = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{lookup}\,(\lambda t.x_{s,t})) = \textbf{lookup}\,(\lambda r.x_{r,r})$$

$$\textbf{act}_m\,(\textbf{act}_n\,x) = \textbf{act}_{mn}\,x \qquad \textbf{act}_m\,\textbf{lookup}\,(\lambda s.x_s) = \textbf{lookup}\,(\lambda r.\,\textbf{act}_m\,x_{r\cdot m})$$

$$\textbf{act}_e\,x = x \qquad \textbf{lookup}\,(\lambda s.\,\textbf{act}_{m_s}\,x_s) = \textbf{lookup}\,(\lambda s.\,\textbf{act}_{n_s}\,x_s)$$

provided for all s, $s \cdot m_s = s \cdot n_s$

Back to Normal State

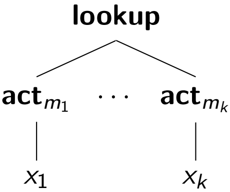
$$\begin{aligned}
\mathbf{lookup}\,(\lambda s.x) &= x & \mathbf{lookup}\,(\lambda s.\,\mathbf{lookup}\,(\lambda t.x_{s,t})) &= \mathbf{lookup}\,(\lambda r.x_{r,r}) \\
\mathbf{act}_m\,(\mathbf{act}_n\,x) &= \mathbf{act}_{mn}\,x & \mathbf{act}_m\,\mathbf{lookup}\,(\lambda s.x_s) &= \mathbf{lookup}\,(\lambda r.\,\mathbf{act}_m\,x_{r\cdot m}) \\
\mathbf{act}_e\,x &= x & \mathbf{lookup}\,(\lambda s.\,\mathbf{act}_{m_s}\,x_s) &= \mathbf{lookup}\,(\lambda s.\,\mathbf{act}_{n_s}\,x_s) \\
& & & \text{provided for all } s,\ s\cdot m_s = s\cdot n_s
\end{aligned}$$

To recover state:

- Define associative operation on S: $s_1 \cdot s_2 := s_2$
- Lift to $\mathbb{M}_{\mathbf{ow}} := \langle \{e\} + S, \cdot \rangle$, *overwrite monoid.*
- Global state is theory for following action over S:

$$\langle s_1, m \rangle \mapsto s_1 \cdot m$$

Normal Form



Free Model: Generalised State Monad

Orbit

For a monoid action $\langle M, S, \cdot \rangle$ and $s \in S$, the *orbit* of s is

$$sM := \{s \cdot m | m \in M\}$$

Generalised State Monad:

$$A \mapsto \prod_{s \in S} sM \times A$$

Useful?

Subsumpiton

Global State

Instantiating the overwrite monoid:

$$A \mapsto \prod_{s \in S} s\mathbb{M}_{\mathbf{ow}} \times A = \prod_{s \in S} S \times A \cong S \to S \times A$$

Reader

Instantiating the trivial monoid:

$$A \mapsto \prod_{s \in S} s\{e\} \times A = \prod_{s \in S} \{s\} \times A \cong \prod_{s \in S} A \cong S \to A$$

What about Writer?

Marriage of Monads and Effects

Idea (Wadler):

For a monad T implementing effects $\mathcal{E}$:

Code using effects $\varepsilon \subseteq \mathcal{E}$ lives inside a monad T_ε.

Applications:

- Safety guarantees.
- Effect-dependent optimisations:

```
let x <= M in let y <= M in N
                ≡
let x <= M in let y <= x in N
```

- Modular functional programming.

for M in $T_{\mathbf{lookup}}$ or $T_{\mathbf{upd}}$

Problem: What's T_ε?

Conservative Restriction

Conservative Restriction $\langle \Sigma_1, E_1 \rangle \hookrightarrow \langle \Sigma_2, E_2 \rangle$:

$\Sigma_1 \subseteq \Sigma_2$ and $\forall s, t \in \Sigma_1$-Terms:

$$E_2 \vdash s = t \iff E_1 \vdash s = t$$

Idea

$$\text{Monad } T_\varepsilon \leftrightarrow \text{Conservative restriction of } T \text{ to } \varepsilon$$

State Restrictions

To determine T_ε:
$$\varepsilon := \{\mathbf{lookup}, \mathbf{act}_{m_1}, \ldots, \mathbf{act}_{m_n}\}$$

Define:

Generated Submonoid $[m_1, \ldots, m_n]$:
Finite combinations from $\{m_1, \ldots, m_n\}$.

Restricted Action $(\cdot_\varepsilon)$:
Same as $(\cdot)$, with elements from $[m_1, \ldots, m_n]$.

Characterisation:
T_ε is the Generalised State theory for $(\cdot_\varepsilon)$.

Special Case: $T_{\{\mathbf{lookup}\}}$ is Reader.

State Restrictions

To determine T_ε:

$$\varepsilon := \{\mathbf{act}_{m_1}, \ldots, \mathbf{act}_{m_n}\}$$

Define:

Indistinguishability:

$$m_1 \equiv m_2 \iff \forall s \in S : s \cdot m_1 = s \cdot m_2$$

M_ε:

The quotient $[m_1, \ldots, m_n]/\equiv$.

Characterisation:

T_ε is Writer with M_ε.

Special Case: Writer M is $T_{\{\mathbf{act}_s | s \in S\}}$ for some faithful action.

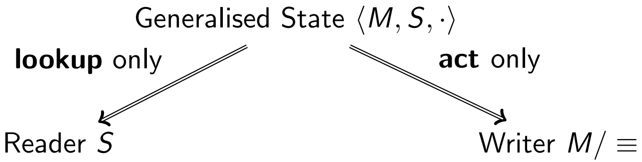
Generalised State Transformers

Let L be a theory with monad T.
The monad for $L \otimes$ Generalised-State is:

$$A \mapsto \prod_{s \in S} T(sM \times A)$$

Conclusion

- Algebraic theory of effects uncovered underlying structure:



- Unified semantic account for global state.
- Useful for programming?

Further work

- Larger work on semantics for type and effect systems.
- Generalising S from a set to CPO (or more general).

Image Sources

- Alfred Leete, *Kitchener Britons*, http://commons.wikimedia.org/wiki/File:Kitchener-Britons.jpg

Generalised State Monad:

$$A \mapsto \prod_{s \in S} sM \times A$$

$$\eta \; : \; a \mapsto \lambda s. \langle s, a \rangle$$

$$\mu \; : \; \lambda s. \left\langle s'_s, \lambda t. \left\langle t'_{s,t}, a_{s,t} \right\rangle \right\rangle \mapsto \lambda r. \left\langle t'_{r,s'_r}, a_{r,s'_r} \right\rangle$$