Algebraic effects and handlers for arrows

Takahiro Sanada [1]

[1]Fukui Prefectural University, Japan

ICFP 2024

Background and contribution

Background 1: algebraic effects and handlers

- The set of programs with effects is an algebra [Plotkin and Power 2001].
- Programmers can implement effects by handlers [Plotkin and Pretnar 2009].

Background 2: arrows

- Arrows are a generalization of monads in Haskell [Hughes 2000][Lindley, Wadler and Yallop 2010][Lindley 2014][Asada 2010].

Contribution: algebraic effects and handlers for arrows

- We reveal the semantic structure of algebraic effects and handlers for arrows.
- Syntax and operational semantics are extracted by the semantic structure.
- Soundness and adequacy theorems are proved.

Semantics

Syntax

Arrows [Hughes 2000]

Arrows are generalization of monads.

The following is the type class of arrows in Haskell.

```
class Arrow a where
  arr :: (x -> y) -> a x y
  (>>>) :: a x y -> a y z -> a x z
  first :: a x y -> a (x, z) (y, z)
```

The following is the type class of monads in Haskell.

```
class Monad m where
  return :: x -> m x
  (>>=) :: m x -> (x -> m y) -> m y
```

Profunctor

A profunctor is a categorical analogue of a relation.

Set theory	Category theory
a function $f \colon A \to B$	a relation $F \colon \mathbb{C} \to \mathbb{D}$
a functor $r \colon B \times A \to 2$	a profunctor $R \colon \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$

- A relation $r \colon A \nrightarrow B$ is a function $r \colon B \times A \to 2$.
- A profunctor $R \colon \mathbb{C} \nrightarrow \mathbb{D}$ is a functor $R \colon \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$.

An example of a profunctor

The hom-functor $\mathbb{I}_{\mathbb{C}} := \mathbb{C}(-,-) \colon \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ is a profunctor $\mathbb{I}_{\mathbb{C}} \colon \mathbb{C} \nrightarrow \mathbb{C}$.

Let **Prof** be the bicategory of categories and profunctors.

Profunctor

A profunctor is a categorical analogue of a relation.

Set theory	Category theory
a function $f\colon A \to B$	a relation $F\colon \mathbb{C} \to \mathbb{D}$
a functor $r\colon B \times A \to 2$	a profunctor $R\colon \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$

- A relation $r\colon A \nrightarrow B$ is a function $r\colon B \times A \to 2$.
- A profunctor $R\colon \mathbb{C} \nrightarrow \mathbb{D}$ is a functor $R\colon \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$.

An example of a profunctor

The hom-functor $\mathrm{I}_{\mathbb{C}} := \mathbb{C}(-,-)\colon \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ is a profunctor $\mathrm{I}_{\mathbb{C}}\colon \mathbb{C} \nrightarrow \mathbb{C}$.

Let **Prof** be the bicategory of categories and profunctors.

Arrows are strong monads [Asada 2010]

$\boxed{\text{Arrows are strong monads in } \textbf{Prof}}$

A strong monad in **Prof** is a profunctor $\mathcal{A} \colon \mathbb{C} \nrightarrow \mathbb{C}$ and 2-cells

$$
\begin{tikzcd}
\mathbb{C} \arrow[r, bend left, "\mathrm{I}_{\mathbb{C}}"{name=U}] \arrow[r, bend right, "\mathcal{A}"'{name=D}] & \mathbb{C}
\arrow[from=U, to=D, phantom, "\Downarrow \eta"]
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathbb{C} \arrow[dr, "\mathcal{A}"] & \\
\mathbb{C} \arrow[ur, "\mathcal{A}"] \arrow[rr, bend right, "\mathcal{A}"'] \arrow[rr, phantom, "\Downarrow \mu"] & & \mathbb{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{C} \times \mathbb{C} \arrow[r, "\mathcal{A} \times \mathrm{I}_{\mathbb{C}}"] \arrow[d, "\otimes_*"'] \arrow[dr, phantom, "\Downarrow \sigma"] & \mathbb{C} \times \mathbb{C} \arrow[d, "\otimes_*"] \\
\mathbb{C} \arrow[r, "\mathcal{A}"'] & \mathbb{C}
\end{tikzcd}
$$

satisfying appropriate axioms.

$$\frac{\eta_{X,Y} \colon \mathrm{I}_{\mathbb{C}} \Rightarrow \mathcal{A}}{\mathbb{C}(X,Y) \xrightarrow{\eta_{X,Y}} \mathcal{A}(X,Y)} \qquad \frac{\mu \colon \mathcal{A} \circ \mathcal{A} \Rightarrow \mathcal{A}}{\mathcal{A}(X,Y) \times \mathcal{A}(Y,Z) \xrightarrow{\mu_{X,Y,Z}} \mathcal{A}(X,Z)}$$

$$\frac{\sigma \colon \otimes_* \circ (\mathcal{A} \times \mathrm{I}_{\mathbb{C}}) \Rightarrow \mathcal{A} \circ \otimes_*}{\mathcal{A}(X,Y) \xrightarrow{\sigma_{X,Y,Z}} \mathcal{A}(X \otimes Z, Y \otimes Z)}$$

Our idea

$\boxed{\text{We construct a strong promonad } \mathcal{A}_\Sigma \text{ on } \mathbf{Set} \text{ in } \mathbf{Prof} \text{ from a signature } \Sigma.}$

- From this semantic structure, a programming language is derived.
  - The language is a extension of the arrow calculus [Lindley, Wadler, Yallop 2010].
  - The language has effects which correspond to arrows and effect handlers.
- Denotational semantics is given by $\mathcal{A}_\Sigma$.

c.f. $\boxed{\mathrm{Term}_\Sigma \text{ is the (strong) monad on } \mathbf{Set} \text{ in } \mathbf{Cat}}$

Construction of $\mathcal{A}_\Sigma$

$\mathcal{A}_\Sigma(X,Y) = \mathrm{Arr}_\Sigma(X,Y)/\!\sim$ is a set of sequences of operations mod $\sim$.

$$\frac{f\colon X \to Y \text{ in } \mathbf{Set}}{X \;—\; f \;—\; Y \quad \in \mathrm{Arr}_\Sigma(X,Y)}$$

$$\frac{\mathsf{op} : \gamma \rightarrowtriangle \delta \in \mathcal{A}_\Sigma}{[\![\gamma]\!] \;—\; \boxed{\mathsf{op}} \;—\; [\![\delta]\!] \quad \in \mathrm{Arr}_\Sigma([\![\gamma]\!], [\![\delta]\!])}$$

$$\frac{\begin{array}{c} X \;—\; \boxed{a} \;—\; Y \quad \in \mathrm{Arr}_\Sigma(X,Y) \\ Y \;—\; \boxed{b} \;—\; Z \quad \in \mathrm{Arr}_\Sigma(Y,Z) \end{array}}{X \;—\; \boxed{a} \;—\; \boxed{b} \;—\; Z \quad \in \mathrm{Arr}_\Sigma(X,Z)}$$

$$\frac{X \;—\; \boxed{a} \;—\; \boxed{b} \;—\; Y \quad \in \mathrm{Arr}_\Sigma(X,Y)}{\begin{array}{lcl} X & —\; \boxed{a} \;—\; \boxed{b} \;— & Y \\ Z & ——————— & Z \end{array} \quad \in \mathrm{Arr}_\Sigma(X \times Z, Y \times Z)}$$

$$—\; \mathrm{id} \;—\; \boxed{a} \;— \;\sim\; —\; \boxed{a} \;—,$$

$$—\; \boxed{a} \;—\; \mathrm{id} \;— \;\sim\; —\; \boxed{a} \;—,$$

$$\vdots$$

$$\Downarrow$$

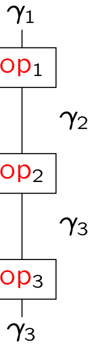
$\mathcal{A}_\Sigma$ is a strong promonad on **Set** in **Prof**!

$$\mathcal{A}_\Sigma : \mathbf{Set} \nrightarrow \mathbf{Set}$$

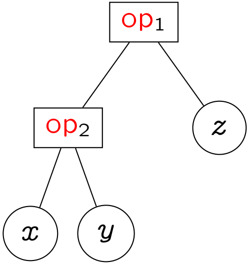
A model is a $\mathbb{C}$-small strong monad $\mathcal{A}$ on a CCC $\mathbb{C}$ in $\mathbb{C}'$-**Prof**, where $\mathbb{C}'$ is a sufficiently cocomplete CCC with a fully faithful cartesian functor $J : \mathbb{C} \to \mathbb{C}'$.

Free algebras for arrows

$\boxed{\mathsf{Arr}_\Sigma(X, Y)}$



$\boxed{\mathsf{Term}_\Sigma(X)}$



Comparison: Arrows and Monads

Arrows

A strong promonad
$\mathcal{A}_\Sigma \colon \mathbf{Set} \nrightarrow \mathbf{Set}$ in **Prof**.

$\mathcal{A}_\Sigma(X, Y)$ is a set of sequences.

An $\mathcal{A}$-algebra is a presheaf $G \colon \mathbb{C}^{op} \to \mathbf{Set}$ with $\alpha \colon \mathcal{A} \circ G \Rightarrow G$.

$$\begin{tikzcd}
\mathbf{Set}(X, Y) \arrow[r, "\eta_{X,Y}"] \arrow[dr, "\phi"'] & \mathcal{A}_\Sigma(X, Y) \arrow[d, dashed, "h"] \\
& G(X)
\end{tikzcd}$$

Interpretation of handlers is given by h.

Monads

A (strong) monad
$\mathrm{Term}_\Sigma \colon \mathbf{Set} \to \mathbf{Set}$ in **Cat**.

$\mathrm{Term}_\Sigma(X)$ is a set of trees.

A $\mathcal{T}$-algebra is a set A with $\alpha \colon \mathcal{T}A \to A$.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\phi"'] & \mathrm{Term}_\Sigma(X) \arrow[d, dashed, "h"] \\
& A
\end{tikzcd}$$

Interpretation of handlers is given by h.

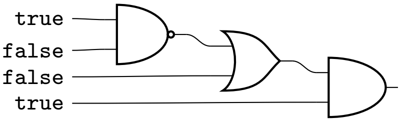
Semantics

Syntax

Logical circuit simulation

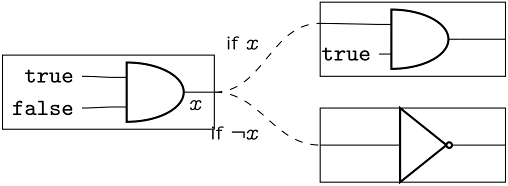
OK: we can write logical circuit using algebraic effects.

$$P = \left(\begin{array}{l} \textbf{let}\, x \Leftarrow \mathrm{NAND}(\mathtt{true}, \mathtt{false})\, \textbf{in} \\ \textbf{let}\, y \Leftarrow \mathrm{OR}(x, \mathtt{false})\, \textbf{in}\, \mathrm{AND}(y, \mathtt{true}) \end{array} \right)$$



FORBIDDEN: arrow does not admit the following conditional expression because an element of free algebra is a sequence, not a tree.

$$Q = \left(\begin{array}{l} \textbf{let}\, x \Leftarrow \mathrm{AND}(\mathtt{true}, \mathtt{false})\, \textbf{in} \\ \textbf{if}\, x\, \textbf{then}\, \mathrm{AND}(x, \mathtt{true})\, \textbf{else}\, \mathrm{NOT}(x) \end{array} \right)$$



Implementation of logic gates by handlers

We can simulate the logic circuit P using handler.

$$\textbf{handle}(\textbf{handle}\, P\, \textbf{with}\, H_1)\,\textbf{with}\, H_2 \rightarrow^* \lfloor \texttt{true} \rfloor.$$

$$H_1 = \left\{ \begin{array}{ll} \mathsf{NAND}, k : \texttt{bool} \rightsquigarrow \texttt{bool} \,;\, z : \texttt{bool} \times \texttt{bool} & \mapsto \\ \textbf{let}\, u \Leftarrow \mathsf{AND}(z)\, \textbf{in} & \\ \textbf{let}\, v \Leftarrow \mathsf{NOT}(u)\, \textbf{in}\, k \bullet v & \\[4pt] \mathsf{OR}, k : \texttt{bool} \rightsquigarrow \texttt{bool} \,;\, z : \texttt{bool} \times \texttt{bool} & \mapsto \\ \textbf{let}\, u \Leftarrow \mathsf{NOT}(\textbf{fst}\, z)\, \textbf{in} & \\ \textbf{let}\, v \Leftarrow \mathsf{NOT}(\textbf{snd}\, z)\, \textbf{in} & \\ \textbf{let}\, w \Leftarrow \mathsf{AND}(\langle u, v \rangle)\, \textbf{in}\, k \bullet w & \end{array} \right\}$$

$$H_2 = \left\{ \begin{array}{l} \mathsf{AND}, k : \texttt{bool} \rightsquigarrow \texttt{bool} \,;\, z : \texttt{bool} \times \texttt{bool} \mapsto \\ k \bullet \left(\begin{array}{l} \textbf{if}(\textbf{fst}\, z) \\ \textbf{then}(\textbf{if}(\textbf{snd}\, z)\, \textbf{then}\, \texttt{true}\, \textbf{else}\, \texttt{false}) \\ \textbf{else}\, \texttt{false} \end{array} \right) \\ \mathsf{NOT}, k : \texttt{bool} \rightsquigarrow \texttt{bool} \,;\, x : \texttt{bool} \mapsto \\ k \bullet (\textbf{if}\, x\, \textbf{then}\, \texttt{false}\, \textbf{else}\, \texttt{true}) \end{array} \right\}$$

Algebraic effects and handlers for arrows

We define a programming language, which is interpreted by a strong monad $\mathcal{A}$ in **Prof**.

We extend arrow calculus (Lindley, Wadler and Yallop, 2011).

Types	$A, B, C, D ::= \beta \mid A \times B \mid A \to B \mid A \rightsquigarrow B$
Contexts	$\Gamma, \Delta ::= x_1 : A_1, \ldots, x_n : A_n$
Terms	$M, N, L ::= x \mid \langle M, N \rangle \mid \mathbf{fst}\, M \mid \mathbf{snd}\, M$
(pure)	$\mid \lambda x : A.M \mid MN \mid \lambda^{\bullet} x : A.P$
Commands	$P, Q, R ::= \lfloor M \rfloor \mid \mathbf{let}\, x \Leftarrow P\, \mathbf{in}\, Q \mid L \bullet M$
(effectful)	$\mid \mathsf{op}(M) \mid \mathbf{handle}\, R\, \mathbf{with}\, H$
Handlers	$H ::= \{ \fatsemi x \mapsto P \} \cup \{ \mathsf{op}, k \fatsemi z \mapsto Q_{\mathsf{op}} \}_{\mathsf{op} \in \Sigma}$

For each operation $\mathsf{op} \in \Sigma$, types γ and δ are assigned.

$$(\mathsf{op} : \gamma \rightarrowtriangle \delta) \in \Sigma$$

Denotational Semantics

(Roughly speaking) A model is a strong monad $\mathcal{A} \colon \mathbb{C} \nrightarrow \mathbb{C}$ in **Prof**. $\mathcal{A}_\Sigma$ is a model.

Typing judgements:

(Pure) terms	$\Gamma \vdash M : A$
(Efefctful) commands	$\Gamma \mathbin{⨾} \Delta \vdash P \mathrel{!} A$
Handlers	$\vdash H : C \Rightarrow D$

Interpretation:

$$[\![\Gamma \vdash M : A]\!] \colon [\![\Gamma]\!] \to [\![A]\!]$$

$$[\![\Gamma \mathbin{⨾} \Delta \vdash P \mathrel{!} A]\!] \colon [\![\Gamma]\!] \to \boxed{\mathcal{A}([\![\Delta]\!], [\![A]\!])}$$

$$\frac{\mathcal{A} \colon \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}}{\mathcal{A} \colon \mathbb{C} \nrightarrow \mathbb{C}}$$

Soundness and adequacy

We can define call-by-value operational semantics:
$M \to M'$ and $P \to P'$.

Soundness

- If $M \to M'$ then $[\![M]\!] = [\![M']\!]$.
- If $P \to P'$ then $[\![P]\!] = [\![P']\!]$.

Adequacy

- If $\diamond \vdash M : \mathbf{unit}$ and $[\![M]\!] = \star \in [\![\mathbf{unit}]\!]$ then $M \to^* \langle\rangle$.
- If $\diamond \,;\, \diamond \vdash P \,!\, \mathbf{unit}$ and $[\![P]\!] = \mathrm{arr}(\star) \in \mathcal{A}_\Sigma(1, [\![\mathbf{unit}]\!])$ then $P \to^* \lfloor\langle\rangle\rfloor$.

The proof of the adequacy is by defining logical relations.