アローに対する
代数的エフェクトとエフェクトハンドラ

眞田 嵩大 [1]

[1] **京都大学 数理解析研究所**

日本ソフトウェア科学会第 40 回大会

背景と本研究の貢献

背景 1：代数的エフェクトとエフェクトハンドラ

- （モナドに対応する）副作用を伴うプログラム全体を**代数**だとみなす考え方
- **エフェクトハンドラ**によって副作用をユーザが自由に実装できる

背景 2：アロー

- Haskell においてモナドの一般化として**アロー**という概念がある
- モナドでは捉えきれない副作用を捉える

貢献：アローに対する代数的エフェクトとエフェクトハンドラ

- アロー版の代数的エフェクトの概念とエフェクトハンドラを考察
- **圏論的な考察**から構文や操作的意味論が導かれる
- 表示的意味論に対する健全性と妥当性を証明
- 定理証明支援系 Agda で言語と型保存・進行定理を形式化

目次

意味論

構文論

応用例

代数的エフェクトの代数とはなにか

代数 $=$ **集合** $+$ **演算**

演算記号の集合 $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \ldots\}$

Σ 代数とは、集合 A とその上に定義された演算 op^A の組のこと。
集合 A 上の n 項演算 op^A とは写像

$$\mathrm{op}^A : A^n \to A$$

のこと

代数の例

- $(\mathbb{N}, 0, +)$: 自然数の加法のなすモノイド
- $(\mathbb{Z}, 0, +)$: 整数の加法のなすモノイド
- $(\mathbb{Z}, 0, +, -(\cdot))$: 整数の加法群
- $(\mathcal{P}(X), \cup, \cap)$: べき集合のなす束
- $(A^*, \epsilon, \bullet)$: 文字列の結合のなすモノイド

代数的エフェクトの代数とはなにか

代数 = 集合 + 演算　　演算記号の集合 $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \ldots\}$

Σ 代数とは、集合 A とその上に定義された演算 op^A の組のこと。
集合 A 上の n 項演算 op^A とは写像

$$\mathrm{op}^A : A^n \to A$$

のこと

代数の例

- $(\mathbb{N}, 0, +)$: 自然数の加法のなすモノイド
- $(\mathbb{Z}, 0, +)$: 整数の加法のなすモノイド
- $(\mathbb{Z}, 0, +, -(\cdot))$: 整数の加法群
- $(\mathcal{P}(X), \cup, \cap)$: べき集合のなす束
- $(A^*, \epsilon, \bullet)$: 文字列の結合のなすモノイド

代数的エフェクトの代数とはなにか

代数 = 集合 + 演算　　演算記号の集合 $\Sigma = \{\mathsf{op}_1, \mathsf{op}_2, \ldots\}$

Σ 代数とは、集合 A とその上に定義された演算 op^A の組のこと。
集合 A 上の n 項演算 op^A とは写像

$$\mathsf{op}^A \colon A^n \to A$$

のこと

代数の例

- $(\mathbb{N}, 0, +)$: **自然数の加法のなすモノイド**
- $(\mathbb{Z}, 0, +)$: **整数の加法のなすモノイド**
- $(\mathbb{Z}, 0, +, -(\cdot))$: **整数の加法群**
- $(\mathcal{P}(X), \cup, \cap)$: **べき集合のなす束**
- $(A^*, \epsilon, \bullet)$: **文字列の結合のなすモノイド**

Take-home message その1

自由代数は木の集まり

考えたい演算記号の集合を $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \dots\}$ とする。
集合 X から自由生成された代数 $\mathrm{Term}_\Sigma(X)$ とは、

- 中間ノードが Σ の要素のいずれかであり、
- 葉ノードが X の要素であるような木の集合のことである。

演算は各 $\mathrm{op} \in \Sigma$ に対して、n 個の木を束ねて新しい木をつくる写像である

$$\mathrm{op} : \mathrm{Term}_\Sigma(X) \times \cdots \times \mathrm{Term}_\Sigma(X) \to \mathrm{Term}_\Sigma(X)$$

$$(t_1, \dots, t_n) \mapsto$$

等式も考えているときは、等式を満たすように必要最小限の分だけ割る

Take-home message その1

自由代数は木の集まり

考えたい演算記号の集合を $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \ldots\}$ とする。

集合 X から自由生成された代数 $\mathrm{Term}_\Sigma(X)$ とは、

- **中間ノードが Σ の要素のいずれかであり、**
- **葉ノードが X の要素であるような木の集合のことである。**

演算は各 $\mathrm{op} \in \Sigma$ に対して、n 個の木を束ねて新しい木をつくる写像である

$$\mathrm{op} : \mathrm{Term}_\Sigma(X) \times \cdots \times \mathrm{Term}_\Sigma(X) \to \mathrm{Term}_\Sigma(X)$$

$$(t_1, \ \ldots, \ t_n) \ \mapsto$$

等式も考えているときは、等式を満たすように必要最小限の分だけ割る

Take-home message その1

自由代数は木の集まり

考えたい演算記号の集合を $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \ldots\}$ とする。
集合 X から自由生成された代数 $\mathrm{Term}_\Sigma(X)$ とは、

- **中間ノードが Σ の要素のいずれかであり、**
- **葉ノードが X の要素であるような木の集合のことである。**

演算は各 $\mathrm{op} \in \Sigma$ に対して、n 個の木を束ねて新しい木をつくる写像である

$$\mathrm{op} : \mathrm{Term}_\Sigma(X) \times \cdots \times \mathrm{Term}_\Sigma(X) \to \mathrm{Term}_\Sigma(X)$$

$$(t_1, \ \ldots, \ t_n) \quad \mapsto \quad \begin{array}{ccccc} & & \mathrm{op} & & \\ & \swarrow & & \searrow & \\ t_1 & \cdots & & & t_n \end{array}$$

等式も考えているときは、等式を満たすように必要最小限の分だけ割る

Take-home message その1

自由代数は木の集まり

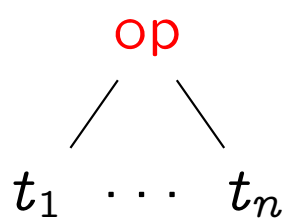
考えたい演算記号の集合を $\Sigma = \{\mathrm{op}_1, \mathrm{op}_2, \dots\}$ とする。
集合 X から自由生成された代数 $\mathrm{Term}_\Sigma(X)$ とは、

- **中間ノードが Σ の要素のいずれかであり、**
- **葉ノードが X の要素であるような木の集合のことである。**

演算は各 $\mathrm{op} \in \Sigma$ に対して、n 個の木を束ねて新しい木をつくる写像である

$$\mathrm{op}\colon\ \mathrm{Term}_\Sigma(X) \times \cdots \times \mathrm{Term}_\Sigma(X) \to \mathrm{Term}_\Sigma(X)$$

$$(t_1, \quad \dots, \quad t_n) \quad \mapsto$$



等式も考えているときは、等式を満たすように必要最小限の分だけ割る

モナドとの関係

数学的にすごくいいやつ。
計算効果を捉えられる

Term_Σ は 2 圏 Cat における Set 上の モナド である

圏と関手のなす 2 圏

集合と写像のなす圏

まず Term_Σ は関手である。

$$\mathsf{Term}_\Sigma : \mathbf{Set} \to \mathbf{Set}$$
$$X \mapsto \mathsf{Term}_\Sigma(X)$$

さらに、モナドの構造、つまり単位 η と掛け算 μ をもつ。

$$\eta_X : X \to \mathsf{Term}_\Sigma(X) \qquad \mu_X : \mathsf{Term}_\Sigma(\mathsf{Term}_\Sigma(X)) \to \mathsf{Term}_\Sigma(X)$$

モナドとの関係

数学的にすごくいいやつ。
計算効果を捉えられる

Term_Σ は 2 圏　Cat　における　Set　上の　モナド　である

圏と関手のなす 2 圏

集合と写像のなす圏

まず Term_Σ は関手である。

$$\mathrm{Term}_\Sigma : \mathbf{Set} \to \mathbf{Set}$$
$$X \mapsto \mathrm{Term}_\Sigma(X)$$

さらに、モナドの構造、つまり単位 η と掛け算 μ をもつ。

$$\eta_X : X \to \mathrm{Term}_\Sigma(X) \qquad \mu_X : \mathrm{Term}_\Sigma(\mathrm{Term}_\Sigma(X)) \to \mathrm{Term}_\Sigma(X)$$

代数的エフェクト [Plotkin and Power, 2001]

プログラム全体の集合 + 副作用を起こす演算によって代数を作る

bool 型のグローバルな状態をもつプログラムを考える。
演算 get で状態に保存されている値を取得できるものとする。
型 A のプログラム全体の集合を $\mathbf{Prog}_A$ とする。

$$\mathsf{get} : \mathbf{Prog}_A \times \mathbf{Prog}_A \to \mathbf{Prog}_A$$
$$(M_{\mathtt{true}}, M_{\mathtt{false}}) \mapsto \mathsf{get}(M_{\mathtt{true}}, M_{\mathtt{false}})$$

$$= \quad \begin{array}{ccc} & \mathsf{get} & \\ \swarrow & & \searrow \\ M_{\mathtt{true}} & & M_{\mathtt{false}} \end{array}$$

OCaml ライクなプログラム let $x \Leftarrow$ get in M は

$$\mathsf{get}(M[\mathtt{true}/x], M[\mathtt{false}/x])$$

の別表記とみなせる。

代数的エフェクト [Plotkin and Power, 2001]

プログラム全体の集合 + 副作用を起こす演算によって代数を作る

bool 型のグローバルな状態をもつプログラムを考える。
演算 get で状態に保存されている値を取得できるものとする。
型 A のプログラム全体の集合を $\mathbf{Prog}_A$ とする。

$$\begin{aligned} \mathsf{get} : \mathbf{Prog}_A \times \mathbf{Prog}_A &\to \mathbf{Prog}_A \\ (M_{\mathtt{true}}, M_{\mathtt{false}}) &\mapsto \mathsf{get}(M_{\mathtt{true}}, M_{\mathtt{false}}) \end{aligned}$$

$$= \quad \begin{array}{ccc} & \mathsf{get} & \\ \diagup & & \diagdown \\ M_{\mathtt{true}} & & M_{\mathtt{false}} \end{array}$$

OCaml ライクなプログラム **let** $x \Leftarrow \mathsf{get}$ **in** M は

$$\mathsf{get}(M[\mathtt{true}/x], M[\mathtt{false}/x])$$

の別表記とみなせる。

エフェクトハンドラ [Plotkin and Pretner, 2009]

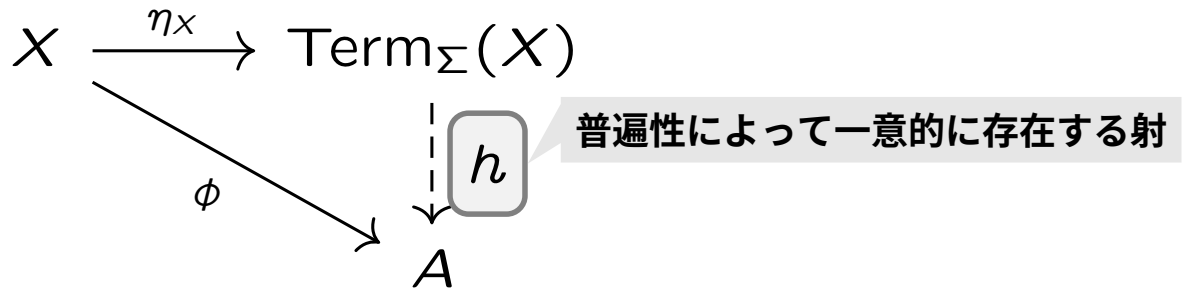
エフェクトハンドラは　準同型射　である

演算を保存する写像 $h\colon A \to B$
$h(\mathrm{op}^A(t_1, \ldots, t_n)) = \mathrm{op}^B(h(t_1), \ldots, h(t_n))$

Σ 代数 A と写像 $\phi\colon X \to A$ があるとき、準同型射

$$h\colon \mathrm{Term}_\Sigma(X) \to A$$
$$x \mapsto \phi(x)$$
$$\mathrm{op}(t_1, \ldots, t_n) \mapsto \mathrm{op}^A(h(t_1), \ldots, h(t_n))$$

が誘導される。これは以下の図式を可換にする唯一の準同型射である。



エフェクトハンドラは準同型射 h を用いて意味が与えられる。

本研究のアイデア

Arr_Σ は双圏　Prof　における Set 上の強モナドである

圏と関係手のなす双圏

となる Arr_Σ を構成する。

- 従来とは異なる計算効果を捉えるプログラミング言語が圏論的に導かれる
  - アローに対する代数的エフェクトとエフェクトハンドラ
- 表示的意味論が Arr_Σ で与えられる
- このプログラミング言語が役に立つ例がある

c.f.

Term_Σ は 2 圏　Cat　における Set 上の（強）モナドである

圏と関手のなす 2 圏

関係手 (profunctor)

関係手とは、関係の圏論版である

集合論	圏論
関数 $f : A \to B$	関手 $F : \mathbb{C} \to \mathbb{D}$
関係 $r : B \times A \to 2$	関係手 $R : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$

- 関係 $r : A \nrightarrow B$ とは、関数 $r : B \times A \to 2$ のこと。
- 関係手 $R : \mathbb{C} \nrightarrow \mathbb{D}$ とは、関手 $R : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ のこと。

関係手の例

hom 関手 $\mathrm{I}_{\mathbb{C}} := \mathbb{C}(-,-) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ は関係手 $\mathrm{I}_{\mathbb{C}} : \mathbb{C} \nrightarrow \mathbb{C}$ である。

関係の合成の類似として、関係手の合成も定義できる。

圏と関係手は双圏 Prof をなす。

関係手 (profunctor)

関係手とは、関係の圏論版である

集合論	圏論
関数 $f : A \to B$	関手 $F : \mathbb{C} \to \mathbb{D}$
関係 $r : B \times A \to 2$	関係手 $R : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$

- 関係 $r : A \nrightarrow B$ とは、関数 $r : B \times A \to 2$ のこと。
- 関係手 $R : \mathbb{C} \nrightarrow \mathbb{D}$ とは、関手 $R : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ のこと。

関係手の例

hom 関手 $\mathrm{I}_{\mathbb{C}} := \mathbb{C}(-,-) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$ は関係手 $\mathrm{I}_{\mathbb{C}} : \mathbb{C} \nrightarrow \mathbb{C}$ である。

関係の合成の類似として、関係手の合成も定義できる。

圏と関係手は双圏 Prof をなす。

アロー [Hughes 2000]

アローは副作用を捉える（モナドとは別の）概念

Haskell **におけるアローの型クラスは以下のとおり**

```
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)
```

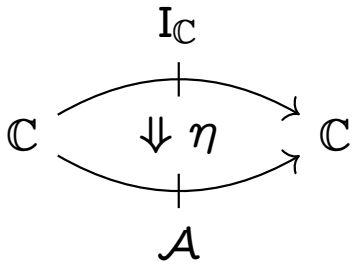
Haskell **におけるモナドの型クラスを思い出しておく**

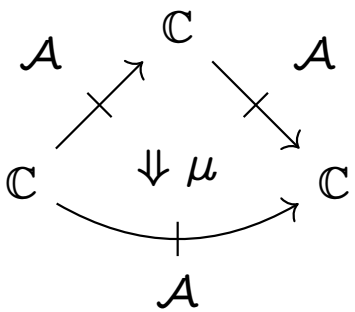
```
class Monad m where
  return :: x -> m x
  (>>=) :: m x -> (x -> m y) -> m y
```

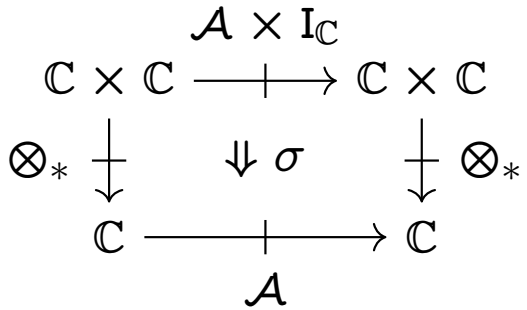
アローは強モナドである [Asada 2010]

アローは双圏 Prof における強モナドである

Prof における強モナドは、関係手 $\mathcal{A} : \mathbb{C} \nrightarrow \mathbb{C}$ と 2-cell たち







であって適当な公理を満たすもののこと。
アローと Prof における強モナドの関係：

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

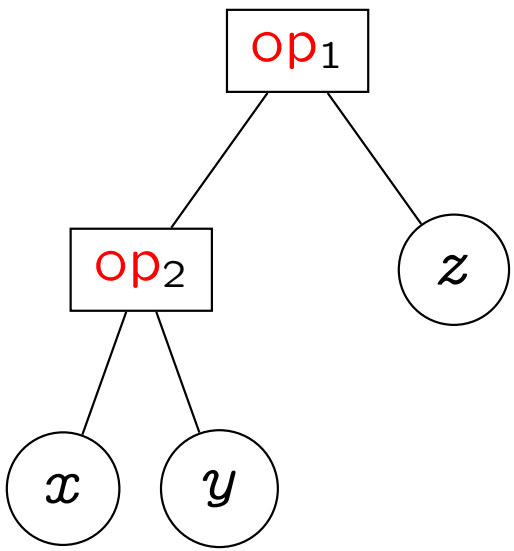
$$\frac{\sigma : \otimes_* \circ (\mathcal{A} \times \mathrm{I}_{\mathbb{C}}) \Rightarrow \mathcal{A} \circ \otimes_*}{\mathcal{A}(X,Y) \xrightarrow{\sigma_{X,Y,Z}} \mathcal{A}(X \otimes Z, Y \otimes Z)}$$

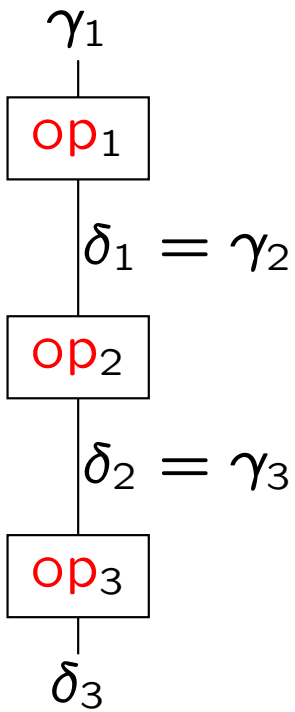
Take-home message その2

アローに対応する自由代数は列の集まり

c.f. (take-home message その 1)

モナドに対応する自由代数は木の集まり





Prof における強モナド $\mathcal{A} \colon \mathbb{C} \nrightarrow \mathbb{C}$ に対する Eilenberg-Moore **圏 $\mathcal{EM}(\mathcal{A})$ への関係手 $a : \mathbf{1} \nrightarrow \mathcal{EM}(\mathcal{A})$ を（アローの）代数だと定義する（モナドの場合の代数は $a \colon \mathbf{1} \to \mathcal{EM}(\mathcal{T})$ であった）。**EM **圏の普遍性や余エンドの計算によって正当化できる。**

Arr_Σ の作り方

$\mathrm{Term}_\Sigma(X)$ のアロー版 $\mathrm{Arr}_\Sigma(X, Y)$ をつくることができる

集合 X, Y に対してクラス $\mathrm{Arr}_\Sigma(X, Y)$ を次で定める。

$$\frac{f \in \mathbf{Set}(X, Y)}{\mathrm{arr}(f) \in \mathrm{Arr}_\Sigma(X, Y)} \qquad \frac{a \in \mathrm{Arr}_\Sigma(X, Y) \quad b \in \mathrm{Arr}_\Sigma(Y, Z)}{a \ggg b \in \mathrm{Arr}_\Sigma(X, Z)}$$

$$\frac{\mathrm{op} : \gamma \nrightarrow \delta \in \Sigma}{\mathrm{op} \in \mathrm{Arr}_\Sigma([\![\gamma]\!], [\![\delta]\!])} \qquad \frac{a \in \mathrm{Arr}_\Sigma(X, Y)}{\mathrm{first}_Z(a) \in \mathrm{Arr}_\Sigma(X \times Z, Y \times Z)}$$

残念なことに $\mathrm{Arr}_\Sigma(X, Y)$ は真クラスになる（$\ggg$ のせい）。

標準形を定義でき、適当な同値関係 $\sim$ を入れると任意の $\mathrm{Arr}_\Sigma(X, Y)$ の要素はある標準形と同値になる。
標準形だけを集めたもの $\mathcal{A}^\circ_\Sigma(X, Y)$ は集合になる。

（Ens-Prof の Set 小な）強モナド $\mathcal{A}_\Sigma : \mathbf{Set} \nrightarrow \mathbf{Set}$ が得られる。

モデルとは ccc $\mathbb{C}$ 上の（$\mathbb{C}'$-Prof の $\mathbb{C}$ 小な）強モナド $\mathcal{A} : \mathbb{C} \nrightarrow \mathbb{C}$ のこと。
ここで $\mathbb{C}'$ は十分余完備な ccc で、忠実充満なカルテシアン関手 $J : \mathbb{C} \to \mathbb{C}'$ がある

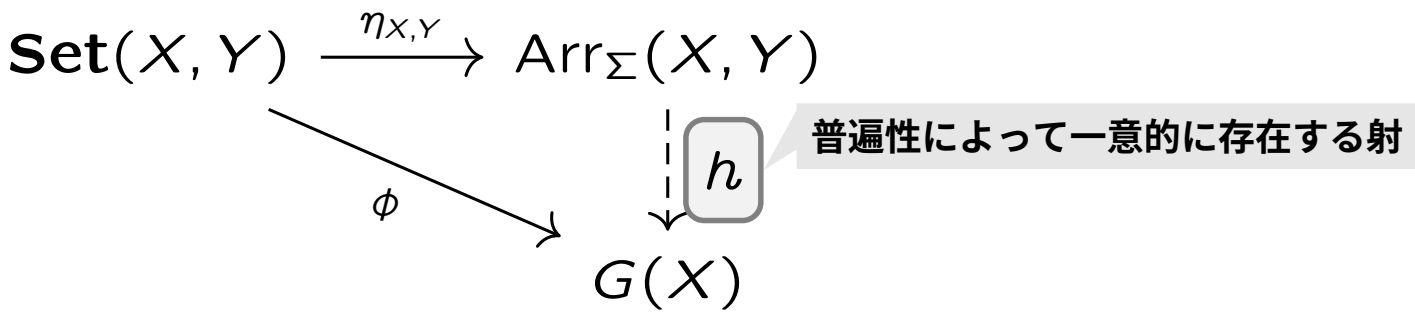
エフェクトハンドラ（アロー版）

エフェクトハンドラは準同型射である

Σ 代数 $G(-)$ と $\phi\colon \mathbf{Set}(-, Y) \to G(-)$ があるとき、準同型射

$$
\begin{aligned}
h\colon\ \mathsf{Arr}_\Sigma(X, Y) &\to G(X) \\
x &\mapsto \phi(x) \\
\mathsf{op} \ggg t &\mapsto \mathsf{op}^G(h(t))
\end{aligned}
$$

が誘導される。これは以下の図式を可換にする唯一の準同型射である。



アローに対するエフェクトハンドラは準同型射 h を用いて意味が与えられる。

意味論

構文論

応用例

アローに対する代数的エフェクトとハンドラ

Prof における強モナド $\mathcal{A}$ で解釈されるような「代数的エフェクト」と「ハンドラ」を持つ言語を作れる

Arrow calculus (Lindley, Wadler and Yallop, 2011) **を拡張する**

型 $A, B, C, D ::= \beta \mid A \times B \mid A \to B \mid A \rightsquigarrow B$

型環境 $\Gamma, \Delta ::= x_1 : A_1, \ldots, x_n : A_n$

項 $M, N, L ::= x \mid \langle M, N \rangle \mid \mathbf{fst}\, M \mid \mathbf{snd}\, M$

(純粋) $\mid \lambda x : A.M \mid MN \mid \lambda^{\bullet} x : A.P$

コマンド $P, Q, R ::= \lfloor M \rfloor \mid \mathbf{let}\, x \Leftarrow P\, \mathbf{in}\, Q \mid L \bullet M$

(計算が生じる) $\mid \mathrm{op}(M) \mid \mathbf{handle}\, R\, \mathbf{with}\, H$

ハンドラ $H ::= \{ \fatsemi x \mapsto P \} \cup \{ \mathrm{op}, k \fatsemi z \mapsto Q_{\mathrm{op}} \}_{\mathrm{op} \in \Sigma}$

シグネチャ Σ の各演算 op には型 γ と δ が割り当てられている。

$$(\mathrm{op} : \gamma \twoheadrightarrow \delta) \in \Sigma$$

表示的意味論

（大雑把に言って）Prof における強モナド $\mathcal{A} : \mathbb{C} \nrightarrow \mathbb{C}$ がモデル

型判断（型付け規則は次のスライド）

純粋な計算	$\Gamma \vdash M : A$
副作用を含んだ計算	$\Gamma \fatsemi \Delta \vdash P \,!\, A$
ハンドラ	$\vdash H : C \Rightarrow D$

強モナド $\mathcal{A}$ の η, μ, σ を使ってコマンド P の解釈を定める

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

$$[\![\Gamma \fatsemi \Delta \vdash P \,!\, A]\!] : [\![\Gamma]\!] \to \mathcal{A}([\![\Delta]\!], [\![A]\!])$$

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

主要な型付け規則

純粋な計算 $\Gamma \vdash M : A$

$$\frac{\Gamma \fatsemi x : A \vdash P \mathbin{!} B}{\Gamma \vdash \lambda^{\bullet} x : A.P : A \rightsquigarrow B} \qquad \ldots$$

副作用を含んだ計算 $\Gamma \fatsemi \Delta \vdash P \mathbin{!} A$

$$\frac{\Gamma, \Delta \vdash M : A}{\Gamma \fatsemi \Delta \vdash \lfloor M \rfloor \mathbin{!} A} \qquad \frac{\Gamma \vdash L : A \rightsquigarrow B \quad \Gamma, \Delta \vdash M : A}{\Gamma \fatsemi \Delta \vdash L \bullet M \mathbin{!} B}$$

$$\frac{\Gamma \fatsemi \Delta \vdash P \mathbin{!} A \quad \Gamma \fatsemi x : A, \Delta \vdash Q \mathbin{!} B}{\Gamma \fatsemi \Delta \vdash \mathbf{let}\, x \Leftarrow P \,\mathbf{in}\, Q \mathbin{!} B} \qquad \frac{\mathrm{op} : \gamma \rightarrowtriangle \delta \in \Sigma \quad \Gamma, \Delta \vdash M : \gamma}{\Gamma \fatsemi \Delta \vdash \mathrm{op}(M) \mathbin{!} \delta}$$

$$\frac{\Gamma \fatsemi \Delta \vdash P \mathbin{!} C \quad \vdash H : C \Rightarrow D}{\Gamma \fatsemi \Delta \vdash \mathbf{handle}\, P \,\mathbf{with}\, H \mathbin{!} D}$$

ハンドラ $\vdash H : C \Rightarrow D$

$$\frac{\Diamond \fatsemi x : C \vdash P \mathbin{!} D \quad (k : \delta \rightsquigarrow D \fatsemi z : \gamma \vdash Q_{\mathrm{op}} \mathbin{!} D)_{(\mathrm{op} : \gamma \rightarrowtriangle \delta) \in \Sigma}}{\vdash \{\fatsemi x \mapsto P\} \cup \{\mathrm{op}, k \fatsemi z \mapsto Q_{\mathrm{op}}\}_{\mathrm{op} \in \Sigma} : C \Rightarrow D}$$

健全性と妥当性

操作的意味論と表示的意味論の間の関係

操作的意味論 $M \to M'$, $P \to P'$ **を** call-by-value **で定めることができる。**

健全性

- $M \to M'$ **ならば** $[\![M]\!] = [\![M']\!]$ **である。**
- $P \to P'$ **ならば** $[\![P]\!] = [\![P']\!]$ **である。**

妥当性

- $\diamond \vdash M : \mathtt{unit}$ **かつ** $[\![M]\!] = \star \in [\![\mathtt{unit}]\!]$ **ならば** $M \to^* \langle\rangle$ **である。**
- $\diamond \fatsemi \diamond \vdash P \,!\, \mathtt{unit}$ **かつ** $[\![P]\!] = \mathrm{arr}(\star) \in \mathcal{A}_\Sigma(1, [\![\mathtt{unit}]\!])$ **ならば** $P \to^* \lfloor\langle\rangle\rfloor$ **である。**

妥当性の証明のためには論理関係 (logical relation) **を適切に定めればよい**

意味論

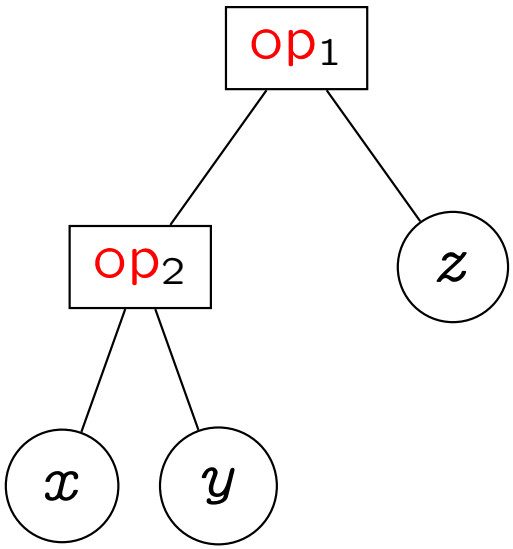
構文論

応用例

応用例：ニューラルネットワークプログラミング

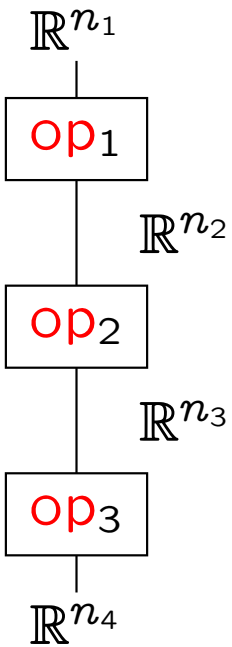
観察：ニューラルネットワークでは、前の層の出力に応じて次の層を動的に変更することはない

モナドに対応する自由代数は木の集まり



前の層の出力に応じて次の層を動的に変更する変なネットワークが書けてしまう

アローに対応する自由代数は列の集まり



層を動的に変更する変なネットワークは書けない

ニューラルネットワークを記述する言語の基礎としてアローが適している

アローハンドラを用いた層の実装

ハンドラがあることによって層の構造の定義と振る舞いの実装を分離できる

1. **まず抽象的に層の構造を定める。**
2. **やりたいことに応じてハンドラを使い分けて実行する**
   - **層の重みの初期化**
     - **適当な初期化**
     - He **初期化**
     - Xavier **初期化**
   - **誤差逆伝播法による重みの学習**
     - **ハンドラでは継続が返した値を利用できる。これを使って逆方向の情報の伝播を実現できる**
   - **推論**
     - ReLU
     - sigmoid
     - tanh

まとめ

- Arrow calculus に代数的エフェクトとハンドラを付け加えて拡張できる。
- （大雑把に言って）Prof における強モナド $\mathcal{A} : \mathbb{C} \nrightarrow \mathbb{C}$ がモデルとなる。
- 具体的なモデルとして強モナド $\mathcal{A}_\Sigma : \mathbf{Set} \nrightarrow \mathbf{Set}$ が構成できる。
- ハンドラは準同型射として解釈される。
- このようにして作られた解釈（表示的意味論）は操作的意味論と整合的である。
  - 健全性
  - 妥当性
- アローに対する代数的エフェクトとハンドラはニューラルネットワークプログラミングに適している。
  - アローに対応する自由代数は列の集まり