エフェクトハンドラは準同型射である

眞田 嵩大

2022 年 12 月 23 日

概要

代数的エフェクト (algebraic effect) を備えたプログラミング言語におけるエフェクトハンドラ (effect handler) は、プログラムのなす代数の間の準同型射のことである。この記事は圏論アドベントカレンダー 23 日目の記事である。

1 導入

代数的エフェクトは Plotkin と Power [PP01] によって導入された、計算効果を代数の演算として理解しようという考え方である。Plotkin と Pretner [PP09] は代数的エフェクトを備えたプログラミング言語にエフェクトハンドラという概念を導入した。エフェクトハンドラはエラーハンドラを一般化した概念である。OCaml のバージョン 5.0.0 以降 [OCa22] ではエフェクトハンドラが導入され、徐々に普及しつつある。

エフェクトハンドラは準同型射のことであるため、プログラマが理解しやすい概念である。ここではこの「エフェクトハンドラは準同型射である」という事実を、ベクトル空間のアナロジーをもとに解説する。

2 ベクトル空間の間の線形写像

体 k を固定する。集合と写像のなす圏を **Set**、k ベクトル空間と線形写像のなす圏を **Vect** と書く。ベクトル空間 V に対して、V の台集合（代数の構造を忘れて単に集合とみなしたもの）を $|V|$ と書くことにする。同様に線形写像（ベクトル空間の間の準同型射）$f\colon V \to W$ に対して、f を単に台集合の間の写像だとみなしたものを $|f|\colon |V| \to |W|$ と書く。$|-|$ は関手 $\mathbf{Vect} \to \mathbf{Set}$ となる。

V, W を k ベクトル空間とする。V の基底 $(\eta_1, \ldots, \eta_n)$ をひとつ固定する。各 η_i $(i = 1, \ldots, n)$ に対して W のベクトル $\phi_i \in W$ を選べば、線形写像 $\tilde{\phi}\colon V \to W$ であって各 i について

$$|\tilde{\phi}|(\eta_i) = \phi_i \tag{1}$$

となるものが一意的に定まるのであった。より具体的に書けば

$$\tilde{\phi}(c_1\eta_1 + \cdots + c_n\eta_n) = c_1\phi_1 + \cdots + c_n\phi_n \tag{2}$$

である。このことを集合の間の写像 $\phi\colon \{1, \ldots, n\} \to |W|$ が線形写像 $\tilde{\phi}$ に**持ち上がる**という。図式で書けば以下のようになる。ここで図式の左側の三角の可換図式は **Set** の図式であり、右側は **Vect** の図式である。

$$\begin{array}{ccc} \{1,\ldots,n\} & \xrightarrow{\eta} & |V| \\ & \searrow^{\phi} & \downarrow{\scriptstyle |\tilde{\phi}|} \\ & & |W| \end{array} \qquad \begin{array}{c} V \\ \downarrow{\scriptstyle \tilde{\phi}} \\ W \end{array} \tag{3}$$

$$\frac{x:\tau \in \Gamma}{\Gamma \vdash x:\tau}\ \text{T-Var} \quad \frac{\Gamma, x:\tau \vdash \lambda x.t:\sigma}{\Gamma \vdash \lambda x.t:\tau\to\sigma}\ \text{T-Abs} \quad \frac{\Gamma \vdash t:\tau\to\sigma \quad \Gamma\vdash s:\tau}{\Gamma\vdash ts:\sigma}\ \text{T-App}$$

$$\frac{c:\tau}{\Gamma\vdash c:\tau}\ \text{T-Const} \quad \frac{}{\Gamma\vdash ():\mathtt{Unit}}\ \text{T-Unit} \quad \frac{(\mathsf{op}:\gamma \twoheadrightarrow \delta)\in\Sigma \quad \Gamma\vdash t:\gamma}{\Gamma\vdash \mathsf{op}(t):\delta}\ \text{T-Op}$$

図 1　代数的エフェクトを持った型システム

$$(\lambda x.t)v \to t[v/x] \qquad \frac{t\to t'}{\mathcal{E}[t]\to\mathcal{E}[t']}$$

図 2　操作的意味論

なぜこのような現象が起きるのかと言えば、V は基底 $(\eta_1,\dots,\eta_n)$ から自由に生成されているため、**自由代数としての普遍性**を持つからである。圏論的に言えば、η が普遍射なのである。

エフェクトハンドラは上の図式でいうところの $\tilde{\phi}$（または $|\tilde{\phi}|$）に相当する。

3　代数的エフェクト

代数的エフェクトは、計算効果をモナドで捉えるというよりも、**代数**として捉えようという考え方である。つまり計算効果を引き起こす操作をプログラムに対する演算として解釈するのである。演算が満たすべき等式を指定することで様々な計算効果を推論したり柔軟に組み合わせることが可能になる。一方、原理的に演算として捉えられないような計算効果（例えばランクを持たないモナドに対応する計算効果など）は代数的エフェクトの枠組みでは扱えない。

代数的エフェクトを持つ型システム（エフェクトシステム）を次で定める。

型	$\sigma,\tau ::= \mathtt{Bool} \mid \mathtt{Int} \mid \mathtt{Unit} \mid \tau\to\sigma$
シグネチャ	$\Sigma = \{(\mathsf{op}_i:\gamma_i \twoheadrightarrow \delta_i)\}_{i\in I} \quad \gamma_i,\delta_i \in \{\mathtt{Bool},\mathtt{Int},\mathtt{Unit}\}$
型文脈	$\Gamma ::= x_1:\tau_1,\dots,x_n:\tau_n$
項	$s,t ::= x \mid \mathtt{true} \mid \mathtt{false} \mid n \mid () \mid \lambda x.t \mid st \mid \mathsf{op}(t)$
値	$u,v ::= x \mid \mathtt{true} \mid \mathtt{false} \mid n \mid () \mid \lambda x.t$

ここで、x, op はそれぞれ変数、演算記号を表す（メタ）変数である。また、$(\lambda x.t)s$ のことを $\mathbf{let}\, x \leftarrow s\, \mathbf{in}\, t$ と表記することもある。型付け規則を図 1 で定める。また評価文脈を $\mathcal{E} ::= [-] \mid \mathcal{E}t \mid v\mathcal{E} \mid \mathsf{op}(\mathcal{E})$ と定義して操作的意味論を図 2 で定める。

演算記号 $(\mathsf{op}:\gamma \twoheadrightarrow \delta)\in\Sigma$ がどのようにして代数の演算だと思えるかということを述べる。今 $(\mathsf{choice}:\mathtt{Unit}\twoheadrightarrow\mathtt{Bool})\in\Sigma$ が非決定的な選択を表す演算だとしよう。このとき choice はプログラム全体の集合上に定義された「Bool 引数」の演算であるとみることができる。プログラム $\mathbf{let}\, x \leftarrow \mathsf{choice}()\, \mathbf{in}\, s$ は、choice() が true を返すか false を返すかに応じて、$s[\mathtt{true}/x]$ が実行されるか $s[\mathtt{false}/x]$ が実行されるかに分岐することを意図している。つまり choice() は true, false : Bool で添え字付けられたプログラムの族 $\{s[\mathtt{true}/x], s[\mathtt{false}/x]\}$ を受け取り $\mathbf{let}\, x \leftarrow \mathsf{choice}()\, \mathbf{in}\, s$ というプログラムを返している代数の演算と見なせる。一般に演算記号 $(\mathsf{op}:\gamma\twoheadrightarrow\delta)\in\Sigma$ は γ をパラーメータとした δ 引数のプログラム上の演算である。こ

の方法により τ 型のプログラム全体の集合に自由 Σ 代数の構造が定まる。これが代数的エフェクトの考え方である。

ベクトル空間との対応付けを行おう。プログラム $\vdash t : \tau$ について、τ がベクトル空間に対応しており[*1]、t がそのベクトルに対応する。プログラムの計算効果は、ベクトル空間における代数構造、つまりベクトルの加法と k 作用に相当する。ベクトル空間の基底つまり生成元に対応するものは、計算効果を含まない**純粋**なプログラム（値）全体の集合である。まとめると図 3 となる。

ベクトル空間の圏 $\mathbf{Vect}$	プログラムの圏
ベクトル空間 V	型 τ
ベクトル $v \in V$	プログラム $\vdash t : \tau$
加法と k 作用	計算効果 $(\mathsf{op} : \gamma \twoheadrightarrow \delta) \in \Sigma$
V の基底	τ 型を持つ値全体の集合

図 3　線形代数の概念とプログラムの概念の対応

4　ハンドラ

まずはハンドラを今考えている言語に追加する。

$$\begin{array}{ll} \text{項} & s, t ::= \cdots \mid \mathbf{handle}\, t\, \mathbf{with}\, h \\ \text{ハンドラ} & h ::= \{x \mapsto s\} \cup \{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{\mathsf{op} \in \Sigma} \end{array}$$

新たに追加するハンドラと項の型付け規則は以下の通りである。

$$\frac{\Gamma, x : \sigma \vdash s : \tau \quad \{\Gamma, x : \gamma, k : \delta \to \tau \vdash s_{\mathsf{op}} : \tau\}_{(\mathsf{op} : \gamma \twoheadrightarrow \delta) \in \Sigma}}{\Gamma \vdash^{\mathrm{h}} \{x \mapsto s\} \cup \{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{\mathsf{op} \in \Sigma} : \sigma \Rightarrow \tau}\ \text{T-Handler}$$

$$\frac{\Gamma \vdash t : \sigma \quad \Gamma \vdash^{\mathrm{h}} h : \sigma \Rightarrow \tau}{\Gamma \vdash \mathbf{handle}\, t\, \mathbf{with}\, h : \tau}\ \text{T-Handle}$$

操作的意味論にはハンドラに関する簡約が追加される。$h = \{x \mapsto t\} \cup \{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{\mathsf{op} \in \Sigma} : \sigma \Rightarrow \tau$ として、

$$\mathbf{handle}\, v\, \mathbf{with}\, h \to t[v/x] \tag{4}$$

$$\mathbf{handle}\, \mathcal{E}[\mathsf{op}(v)]\, \mathbf{with}\, h \to s_{\mathsf{op}}[v/x, \lambda y.\, \mathbf{handle}\, \mathcal{E}[y]\, \mathbf{with}\, h/k] \tag{5}$$

と定める。

この謎めいた操作的意味論をベクトル空間の線形写像の議論を思い出して理解しよう。ハンドラ $h = \{x \mapsto t\} \cup \{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{\mathsf{op} \in \Sigma} : \sigma \Rightarrow \tau$ の型付けが $\vdash^{\mathrm{h}} h : \sigma \Rightarrow \tau$ となっているとする。

ハンドラ h の $\{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{(\mathsf{op} : \gamma \twoheadrightarrow \delta) \in \Sigma}$ の部分を理解しよう。型付け規則 T-Handler を見ると、この部分は各 $\mathsf{op} : \gamma \twoheadrightarrow \delta$ について次の型付けを持つ。

$$x : \gamma, k : \delta \to \tau \vdash s_{\mathsf{op}} : \tau$$

$x : \gamma$ は単なるパラメータだから簡単のため無視してしまおう。すると

$$k : \delta \to \tau \vdash s_{\mathsf{op}} : \tau$$

[*1] より正確に言えば $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$ の型 τ がベクトル空間に対応する。型文脈に現れる型 τ_i は対応しない。

という型付けが得られる。$k : \delta \to \tau$ は δ 個の τ 型のプログラムとみなせる。すると s_{op} の型付けは「δ 個の τ 型のプログラムを受け取って τ 型のプログラムを返す」と読むことができる。このことは τ 型のプログラムの集合の上に**自由でない** Σ 代数の構造をプログラマが勝手に決めているということである。まとめると、$\{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{(\mathsf{op}:\gamma\to\delta)\in\Sigma}$ によって τ 型のプログラムが通常の方法（自由 Σ 代数）とは異なる Σ 代数の構造が入る。このような Σ 代数の構造が入った τ 型のプログラム全体の集合が式 (3) におけるベクトル空間 W に対応している。

ハンドラ h の $\{x \mapsto t\}$ の部分は σ 型の値 v に対して $t[v/x]$ という τ 型のプログラムを割り当てている。ここで図 3 を思い出すと、プログラムにおける値はベクトル空間の基底ベクトルに相当するのであった。するとこのような割り当て $v \mapsto t[v/x]$ は、式 (3) における W のベクトルの選択 $\phi : i \mapsto \phi_i$ に相当していることになる。

さて、これでハンドラを準同型射として理解する準備が整った。ハンドラ $\vdash^{\mathrm{h}} h : \sigma \Rightarrow \tau$ は、σ 型のプログラム全体の自由 Σ 代数から、τ 型のプログラム全体の集合に $\{\mathsf{op}, x, k \mapsto s_{\mathsf{op}}\}_{(\mathsf{op}:\gamma\to\delta)\in\Sigma}$ で定まる Σ 代数への準同型射である。ハンドラの操作的意味論のうち式 (4) は式 (1) と対応している。また式 (5) は式 (2) と対応している。後者の対応をもう少し詳しく述べる。式 (5) のパラメータの部分を無視してしまうことにすると次のように書き直される。

$$\mathbf{handle}\,\mathcal{E}[\mathsf{op}]\,\mathbf{with}\,h \to s_{\mathsf{op}}[(\lambda y.\,\mathbf{handle}\,\mathcal{E}[y]\,\mathbf{with}\,h)/k] \tag{6}$$

$\mathsf{op} : \delta$ は δ 引数の演算であり、今の場合は「δ 個の σ 型のプログラム」$\mathcal{E}[-]$ を引数として受け取り $\mathcal{E}[\mathsf{op}]$ というひとつの σ 型のプログラムを返している。分かりやすさのために $\mathcal{E}[\mathsf{op}]$ のことを $\mathsf{op}(\lambda y.\mathcal{E}[y])$ と書いてしまう。また、$\mathbf{handle}\,t\,\mathbf{with}\,h$ を $h\langle t\rangle$ と書くことにする。すると、式 (6) はさらに以下のように書き直される。

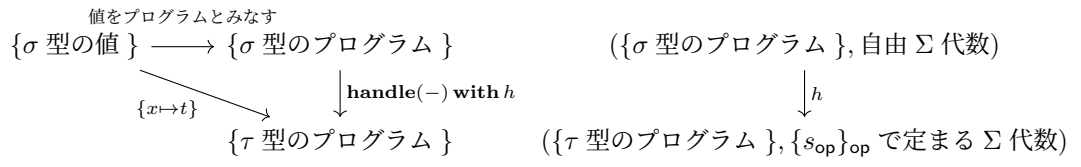
$$h\langle \mathsf{op}(\lambda y.\mathcal{E}[y])\rangle \to s_{\mathsf{op}}[(\lambda y.h\langle\mathcal{E}[y]\rangle)/k] \tag{7}$$

一方、ベクトル空間 V の加法を $+_V$、W の加法を $+_W$ とすると式 (2) から次の式（線形性）が成り立つ。

$$\tilde{\phi}(v_1 +_V v_2) = \tilde{\phi}(v_1) +_W \tilde{\phi}(v_2) \tag{8}$$

式 (7) と式 (8) はそれぞれ自由代数の演算 op と $+_V$ を s_{op} と $+_W$ で解釈しているのである。

このようにして、謎めいていたハンドラの操作的意味論の式 (4) と式 (5) は単に準同型射の計算をしているのだということが分かった。式 (3) に対応するプログラムの可換図式は以下のようになる。



参考文献

[OCa22] Release of ocaml 5.0.0, 2022. `https://ocaml.org/news/ocaml-5.0`.

[PP01] Gordon D. Plotkin and John Power. Adequacy for algebraic effects. In Furio Honsell and Marino Miculan, editors, *FoSSaCS 2001*, volume 2030 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2001.

[PP09] Gordon D. Plotkin and Matija Pretnar. Handlers of algebraic effects. In Giuseppe Castagna, editor, *ESOP 2009*, volume 5502 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2009.