

計算効果の数理モデル

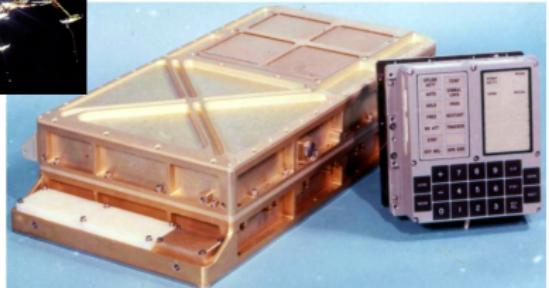
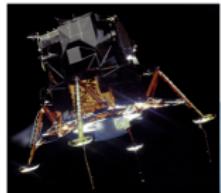
勝股 審也

国立情報学研究所

2022/08/30



副作用を起こすプログラムは至るところで動いている



左: 着陸船イーグルとアポロ誘導コンピュータ (1966-)



プログラミング言語の意味論(数学的モデル)

```
(* separate unit horn clauses from the horn set *)
let rec separate_a_unit hs =
  match hs with
  [] => raise Not_found
  | [x]:lhs => (x, hs')
  | h:hs' => let x, mh = separate_a_unit hs' in (x, h:mh)

(* do the complete unit propagation of a given horn clause set *)
let rec completely_propagate h =
  try
    let v, h' = separate_a_unit h in
    let h'' = propagate v h' in
    let ru, rp = completely_propagate h'' in v::ru, rp
  with
  Not_found => [], h

(* Distributing the disjunction *)
let mapappend f l a =
  let rec loop l' =
    match l with
    [] => a
    | x::l' => f x::loop l' in
  loop l
```

プログラム

通常の実行



意味論



Furthermore, we say that images are *stable* if for each pullback square on the left,

$$\begin{array}{ccc} \bullet & \xrightarrow{j} & \bullet \\ \downarrow v & & \downarrow u \\ \bullet & \xrightarrow{w} & J \\ \downarrow & & \downarrow \\ I & \xrightarrow{u} & J \end{array}$$

the right square is also a pullback, where $\text{Im}(v) \rightarrow \text{Im}(u)$ is the morphism obtained by the universal property of image factorisation of v .

A category is *regular* if it has finite limits and stable images.

Proposition 2.4.19 Let \mathcal{C} be a category with finite limits. Then \mathcal{C} is a regular category if and only if $\text{pc}_{\mathcal{C}}$ has simple coproducts satisfying Frobenius.

PROOF See [Jac99], theorem 4.4.4. ■

意味論の意義

- プログラム・システムの数学的扱い \implies 形式手法の基礎
- 数学と協調して発展(領域理論、型理論、圏論、...)

計算効果をモデルするために 圏論に由来する構造である**モナド**を使おう

- E. Moggi. Computational Lambda-Calculus and Monads. In Proc. LICS, 1989.
- E. Moggi. Notions of computation and monads. Information and Computation 93 (1), 1991

Definition. A monad $T = \langle T, \eta, \mu \rangle$ in a category X consists of a functor $T: X \rightarrow X$ and two natural transformations

$$\eta: I_X \rightarrow T, \quad \mu: T^2 \rightarrow T \tag{1}$$

which make the following diagrams commute

$$\begin{array}{ccc} T^3 & \xrightarrow{T\mu} & T^2 \\ \downarrow \mu T & & \downarrow \mu \\ T^2 & \xrightarrow{\mu} & T, \end{array} \quad \begin{array}{ccccc} IT & \xrightarrow{\eta T} & T^2 & \xleftarrow{T\eta} & TI \\ \parallel & & \downarrow \mu & & \parallel \\ T & = & T & = & T. \end{array} \tag{2}$$

Saunders Mac Lane. Categories for the Working Mathematician (2nd ed). Springer, 1998.

プログラムの素朴な数学的モデル

```
nat f(nat n) {  
    if (n < 2) {  
        return 1;  
    } else {  
        return f(n-1)+f(n-2);  
    }  
}
```

$$[f] : \mathbb{N} \rightarrow \mathbb{N}$$

引数	返り値
0	1
1	1
2	2
3	3
⋮	⋮

$$[g] : \mathbb{N} \rightarrow \mathbb{N}$$

```
nat g(nat n) {  
    printf("Hello!");  
    return 0;  
}
```

引数	返り値
0	0
1	0
2	0
⋮	⋮

モナドによる計算効果の表現

$$f : \text{nat} \rightarrow \text{nat} \quad \Rightarrow \quad \llbracket f \rrbracket : \mathbb{N} \rightarrow \textcolor{blue}{M}(\mathbb{N})$$

値域をモナドでふくらませ、関数が返り値に加えて途中の計算効果について語れるようにする。

モナドによる計算効果の表現

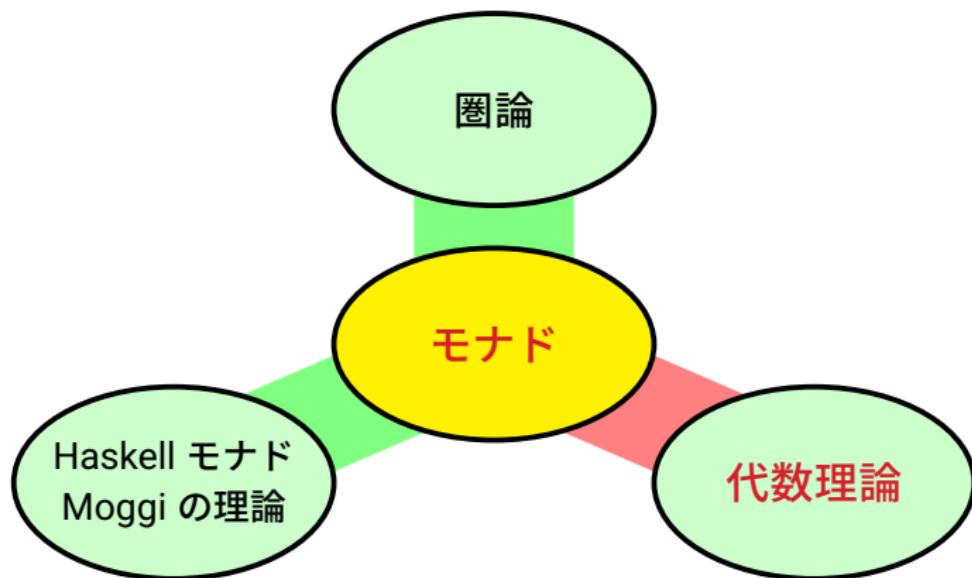
$$f : \text{nat} \rightarrow \text{nat} \quad \Rightarrow \quad \llbracket f \rrbracket : \mathbb{N} \rightarrow \textcolor{blue}{M}(\mathbb{N})$$

値域をモナドでふくらませ、関数が返り値に加えて途中の計算効果について語れるようにする。

決定的な瞬間は Moggi の博士論文の口頭試問の時に来た。... 話題が将来課題に移った際、Moggi は計算の概念をモナドを用いてモデルするアイデアを披露した。この特段エレガントな考えは即座に Hyland に感銘を与えた。...

(抄訳) M. Hyland and J. Power.
The Category Theoretic Understanding of
Universal Algebra: Lawvere Theories and Monads
ENTCS 172, April, 2007, pp. 437–458

本講演のアプローチ



Part I

代数的構造

代数的構造の例: モノイド

集合 X

関数 $e : 1 \rightarrow X$
 $(*) : X^2 \rightarrow X$

の組で以下の条件を満たすものを**モノイド**と呼ぶ。

$$x * e = x, \quad e * x = x, \quad (x * y) * z = x * (y * z)$$

他にも(可換)群、(可換)環、結び半束、分配束、ベクトル空間、ブール代数、ハイティング代数、…

代数的構造と計算効率にどのような関係が?

代数的構造の例: ニモイド [Melliès14][Plotkin&Power02]

集合 X

関数 $r : X^2 \rightarrow X$

$w_0, w_1 : X \rightarrow X$

の組で以下を満たすものをニモイドと呼ぶ。

$$r(w_0(x), w_1(x)) = x, \quad w_i(w_j(x)) = w_j(x), \quad w_i(r(x_0, x_1)) = w_i(x_i)$$

関数の意図: 1ビットメモリの読み書きという計算効果

$r(x_0, x_1)$ メモリを読み、0 の時 x_0 、1 の時 x_1 を行う

$w_i(x)$ メモリに i を書いて x を行う

$r(w_1(x), w_0(x))$ メモリを読み、0 の時メモリに 1 を書いて x を行い、1 の時メモリに 0 を書いて x を行う。
⇒ メモリのビットを反転して x をする

Paul-André Melliès: Local States in String Diagrams. RTA-TLCA 2014: 334-348

Gordon D. Plotkin, John Power: Notions of Computation Determine Monads. FoSSaCS 2002: 342-356

ニモイドの練習問題

(X, r, w_0, w_1) をニモイドとする。

補題

$w_0(x) = w_0(y)$ かつ $w_1(x) = w_1(y)$ ならば $x = y$ 。

証明: $x = r(w_0(x), w_1(x)) = r(w_0(y), w_1(y)) = y$ 。

補題

$r(r(x, y), r(z, w)) = r(x, w)$ 。

上の補題より $w_0(r(r(x, y), r(z, w))) = w_0(r(x, w))$ と
 $w_1(r(r(x, y), r(z, w))) = w_1(r(x, w))$ を示せば十分。1つ目の等式は

$$w_0(r(r(x, y), r(z, w))) = w_0(r(x, y)) = w_0(x) = w_0(r(x, w))$$

2つ目の等式も同様に示せる。

計算効果が誘導する代数的構造

計算効果を起こす命令の代数的性質に着目することで代数的構造が得られる

例: 非決定的選択

- 関数: $x \vee y$: 「 x か y のどちらかを行う」
- 公理: $x \vee x = x$, $x \vee y = y \vee x$, $(x \vee y) \vee z = x \vee (y \vee z)$

例: 確率的選択 (barycentric calculus [Stone49], convex space [Fritz09])

- 関数: $x \oplus_p y$ ($0 \leq p \leq 1$): 「確率 p で x を、確率 $1-p$ で y を行う」
- 公理: $x \oplus_p x = x$, $x \oplus_1 y = x$, $x \oplus_p y = y \oplus_{1-p} x$,
 $(x \oplus_p y) \oplus_q z = x \oplus_{?_1} (y \oplus_{?_2} z)$ ($pq < 1$)

問: 上の $?_1, ?_2$ をいくつにするのが妥当か? p, q により表わせ。

代数的構造の定義のテンプレート

集合 X

関数 $f_1 : X^{a_1} \rightarrow X$

$f_2 : X^{a_2} \rightarrow X$

\vdots

\vdots

の組で以下を満たすものを... と呼ぶ。

$$L_1 = R_1, \quad L_2 = R_2, \quad \dots$$

(ただし L_i, R_i は f_i と変数のみを用いて作られる式)

代数的構造の定義のテンプレート

集合 X

関数 $f_1 : X^{a_1} \rightarrow X$

$f_2 : X^{a_2} \rightarrow X$

\vdots

\vdots

の組で以下を満たすものを... と呼ぶ。

$$L_1 = R_1, \quad L_2 = R_2, \quad \dots$$

(ただし L_i, R_i は f_i と変数のみを用いて作られる式)

代数理論の表現

定義: 代数理論の表現

ランク付きアルファベット Σ と等式公理の集合 E の組を代数理論の表現と呼ぶ。

$$\begin{aligned}\Sigma &= \{f_1 : a_1, f_2 : a_2, \dots\} : O \rightarrow \mathbb{N} \\ E &\subseteq \{(n, L, R) \mid n \in \mathbb{N}, L, R \in T_\Sigma\{x_1, \dots, x_n\}\}\end{aligned}$$

ここで $T_\Sigma X$ は X 変数 Σ 式の集合:

$$T_\Sigma X \ni t ::= x \mid f_i(\overbrace{t, \dots, t}^{a_i}) \quad (x \in X, f_i : a_i \in \Sigma)$$

定義

(Σ, E) が定める代数的構造の定義に従うものを (Σ, E) 代数と呼ぶ。

代数理論の表現の例: モノイド

集合 X

関数 $e : 1 \rightarrow X$
 $(*) : X^2 \rightarrow X$

の組で以下を満たすものをモノイドと言う。

$$x * e = x, \quad e * x = x, \quad (x * y) * z = x * (y * z)$$

例: モノイドの代数理論の表現とその代数

- $\Sigma_m = \{e : 0, * : 2\}$
- $E_m = \{(1, x_1 * e, x_1), (1, e * x_1, x_1), (3, \dots)\}$

$(X, e, (*))$ は (Σ_m, E_m) 代数 $\iff (X, e, (*))$ は モノイド

Part II

代数理論の表現からモナドへ

(Σ, E) 式

X を集合とする。

以下を満たす最小の同値関係を $\equiv_X \subseteq (T_\Sigma X)^2$ とおく。

$$\frac{t_1 \equiv_X t'_1 \quad \cdots \quad t_a \equiv_X t'_a \quad f : a \in \Sigma}{f(t_1, \dots, t_a) \equiv_X f(t'_1, \dots, t'_a)} \text{ 合同性}$$

$$\frac{t_1, \dots, t_n \in T_\Sigma X \quad (n, L, R) \in E}{L[t_1/x_1, \dots, t_n/x_n] \equiv_X R[t_1/x_1, \dots, t_n/x_n]} \text{ 公理}$$

例えば (Σ_m, E_m) の \equiv_X は以下を満たす。

$$(x * (y * z)) * e \equiv_X x * (y * z) \equiv_X (x * y) * z \equiv_X ((e * x) * y) * z$$

以降、 \equiv_X 同値類を **X 变数** (Σ, E) 式と呼ぼう。

(Σ, E) 式の集合

定義: T_{Σ}^E 集合を集合に写す割り当て

集合 X に対し、 X 変数 (Σ, E) 式の集合を返す。

$$T_{\Sigma}^E X \triangleq (T_{\Sigma} X) / \equiv_X$$

定理: (X 上の自由 (Σ, E) 代数)

各 $f : a \in \Sigma$ に対し関数 $[f]$ を以下で定義することで:

$$[f]([t_1]_{\equiv}, \dots, [t_a]_{\equiv}) \triangleq [f(t_1, \dots, t_a)]_{\equiv}$$

$(T_{\Sigma}^E X, [f_1], [f_2], \dots)$ は (Σ, E) 代数となる。

(Σ, E) 式の集合

具体的な (Σ, E) に対し $T_{\Sigma}^E X$ はどうなるか?

命題

代数理論の表現 (Σ, E) が

- ① モノイドのその時 $T_{\Sigma}^E X \simeq X^*$
- ② ニモイドのその時 $T_{\Sigma}^E X \simeq 2 \Rightarrow (X \times 2)$
- ③ 非決定的選択のその時 $T_{\Sigma}^E X \simeq \{U \mid U \subseteq_{fin} X, U \neq \emptyset\}$
- ④ 確率的選択のその時 $T_{\Sigma}^E X \simeq \{\mu : X \text{ 上の有限確率分布}\}$

(Σ, E) がニモイドの代数理論の表現の場合

任意の $t \in T_{\Sigma}^E X$ に対して唯一 $i, j \in 2, x, y \in X$ が存在して
 $t \equiv_X r(w_i(x), w_j(y))$ つまり

$$T_{\Sigma}^E X \cong 2 \times 2 \times X \times X \cong 2 \Rightarrow (2 \times X)$$

(Σ, E) 式のもつ構造: var と sub

以下 X, Y は任意の集合とする。

定義: $\text{var}_X : X \rightarrow T_{\Sigma}^E X$

変数 $x \in X$ に対し、 x からなる式 $[x]_{\equiv}$ を返す。

$$\text{var}_X(x) \triangleq [x]_{\equiv}$$

定義: $\text{sub}_{X,Y} : (X \Rightarrow T_{\Sigma}^E Y) \rightarrow (T_{\Sigma}^E X \Rightarrow T_{\Sigma}^E Y)$

$$\text{sub}_{X,Y}(\theta)(P) \triangleq P[\theta(x)/x]_{x \in X}$$

式 $P \in T_{\Sigma}^E X$ の各変数 $x \in X$ に式 $\theta(x) \in T_{\Sigma}^E Y$ を代入して得られる式を返す。

Haskell の bind と sub を関連づけよう:

$$P \gg= \theta \triangleq \text{sub}_{X,Y}(\theta)(P)$$

var と sub の満たす法則

1. 変数への代入

$$\begin{aligned} sub_{X,Y}(\theta) \circ var_X &= \theta \\ (\iff \forall x \in X . x[\theta(x)/x]_{x \in X} &= \theta(x)) \\ (\iff \forall x \in X . var_X(x) \gg= \theta &= \theta(x)) \end{aligned}$$

2. 変数の代入

$$\begin{aligned} sub_{X,X}(var_X) &= \text{id}_{T_{\Sigma}^E X} \\ (\iff \forall P \in T_{\Sigma}^E X . P[x/x]_{x \in X} &= P) \\ (\iff \forall P \in T_{\Sigma}^E X . P \gg= (\lambda x . var_X(x)) &= P) \end{aligned}$$

3. 連続代入

$$\begin{aligned} sub_{Y,Z}(\theta') \circ sub_{X,Y}(\theta) &= sub_{X,Z}(sub_{Y,Z}(\theta') \circ \theta) \\ (\iff \forall P \in T_{\Sigma}^E X . P[\theta(x)/x]_{x \in X}[\theta'(y)/y]_{y \in Y} &= P[\theta(x)[\theta'(y)/y]_{y \in Y}/x]_{x \in X}) \\ (\iff \forall P \in T_{\Sigma}^E X . (P \gg= \theta) \gg= \theta' &= P \gg= (\lambda x . \theta(x) \gg= \theta')) \end{aligned}$$

圏 Set 上のモナド

定義: 圏 Set 上のモナド (Kleisli トリプル形式)

... は以下の 3 つからなる。

- M : 集合を受け取り集合を返す写像
- 各集合 X に対して定められた関数 $\eta_X : X \rightarrow M(X)$
- 各集合 X, Y に対して定められた関数

$$kl_{X,Y} : (X \Rightarrow M(Y)) \rightarrow (M(X) \Rightarrow M(Y))$$

これらは以下を満たさなければならない (kl の添字は省略)。

$$kl(f) \circ \eta_X = f \quad kl(\eta_X) = \text{id}_{M(X)} \quad kl(g) \circ kl(f) = kl(kl(g) \circ f)$$

モナドは (Σ, E) -式の構成、変数、代入の 3 つを抽象化したもの

定理: 代数理論の表現から作られるモナド

任意の代数理論の表現 (Σ, E) に対し、 $(T_\Sigma^E, \text{var}, \text{sub})$ は圏 Set 上のモナド。

モナドの発祥と名前の由来

モナドの双対であるコモナドの概念は Godement により認識され、初期の頃モナドは triple, dual standard construction などと呼ばれていた。

- R. Godement. Topologie Algébrique et Théorie des Faisceaux. Hermann 1958.

...1966年の夏 (...)、Oberwohlfachで圏論の会合があった。... ある昼食か夕食のとき、Jean Benabou の隣に偶然座ったのだが、彼は私の方を向いて「モナドはどうだろう」というようなことを言った。すこし考えた後、なかなか良いように聞こえると彼にいった（実際そう思った）。それで Jean は一般の聴衆に向けてこれを提案し、同意を得られた。この言葉は当然ながらモノイドを示唆していた。実際モナドは関手圏におけるモノイドである。...

(抄訳) Michael Barr

categories mailing list, 1 Apr 2009 13:13:55 -0500 (EST)

<https://www.mta.ca/~cat-dist/archive/2009/09-4>

計算効果のモデルに用いられる圏 Set 上のモナド

代数理論の表現から作ることができるもの

名前	$\textcolor{blue}{M}(X)$	モデルできる計算効果
Maybe	$X + 1$	計算の発散
例外	$X + E$	例外発生
有限幂集合	$\{U \mid U \subseteq_{fin} X\}$	有限個の非決定的選択
有限確率分布	$\{\mu : X \text{ 上の有限確率分布}\}$	有限個の確率的選択
有限状態	$S \Rightarrow (X \times S)$	メモリ (S: 有限)
ライター	$C^* \times X$	文字出力 (C: 文字集合)
リスト	X^*	有限個の非決定的選択

代数理論の表現から作れないもの

名前	$\textcolor{blue}{M}(X)$	モデルできる計算効果
幂集合	$\{U \mid U \subseteq X\}$	一般の非決定的選択
状態	$S \Rightarrow (X \times S)$	メモリ (S: 無限)
継続	$(X \Rightarrow R) \Rightarrow R$	継続 ($R \not\cong 1$)

Part III

計算効果のあるプログラムの
モナド的意味論

Moggi の理論: モナドによる計算効果の一般的モデル

計算効果のモナドによるモデル

- E. Moggi. Computational Lambda-Calculus and Monads. In Proc. LICS, 1989.
- E. Moggi. Notions of computation and monads. Information and Computation 93 (1), 1991

計算効果のあるプログラミング言語の集合論的意味を与えるには

- 圏 **Set** 上のモナド T を取り、
- プログラム P を $\llbracket P \rrbracket : X \rightarrow TY$ という関数で解釈

あるいは以下のようにしても良い:

- 代数理論の表現 (Σ, E) を取り、
- プログラム P を $\llbracket P \rrbracket : X \rightarrow T_{\Sigma}^E Y$ という関数で解釈

1ビットの大域メモリのある命令形プログラミング言語

以下の命令が備わった単純な命令形プログラミング言語を考える

$$b = rd() \quad wr_0() \quad wr_1()$$

$$P \triangleq int\ f(int\ x)\{b = rd();\ if\ b\ then\ wr_1();\ else\ wr_0();\ return\ x + b\}$$

このプログラムの意味を与えるため、ニモイドの代数理論の表現を取る：

$$\Sigma = \{r : 2, w_0 : 1, w_1 : 1\}, \quad E = \{\text{ニモイドの公理}\}$$

$r(x_0, x_1)$ メモリを読み、0 の時 x_0 、1 の時 x_1 を行う

$w_i(x)$ メモリに i を書いて x を行う

すると P を以下の関数 $\llbracket P \rrbracket : \mathbb{Z} \rightarrow T_{\Sigma}^E \mathbb{Z}$ で解釈できる

$$\llbracket P \rrbracket(x) \triangleq r(w_1(x), w_0(x+1))$$

入出力のある命令形プログラミング言語

以下の命令が備わった単純な命令形プログラミング言語を考える

$\text{puts}_s() \quad x = \text{getchar}() \quad (s \in \{0, \dots, 255\}^*)$

$P \triangleq \text{int } f(\text{int } x) \{ \text{puts}_{\text{called}}(); y = \text{getchar}(); \text{return } x + y \}$

このプログラムの意味を与えるため、以下の代数理論の表現を取る：

$$\Sigma = \{i : 256\} \cup \{o_c : 1 \mid 0 \leq c \leq 255\}, \quad E = \emptyset$$

$i(x_0, \dots, x_{255})$ 入力 $0 \leq i \leq 255$ を受け取り x_i を実行

$o_c(x)$ c を出力して x を実行

すると P を以下の関数 $\llbracket P \rrbracket : \mathbb{Z} \rightarrow T_{\Sigma}^E \mathbb{Z}$ で解釈できる

$$\llbracket P \rrbracket(x) \triangleq o_c(o_a(\dots o_e(o_d(i(t_0, \dots, t_{255}))) \dots)),, \quad t_y = x + y$$

計算効果のある言語

以下の3つのプログラミング言語に関して計算効果がどのように解釈されるかを見ていく。

- ① 命令形プログラミング言語: 変数更新 (C,Java,...)
- ② λ_c -計算: 値呼び関数型プログラミング言語 (SML,OCaml,...)
E. Moggi. Computational Lambda-Calculus and Monads. In Proc. LICS, 1989.
- ③ λ_{ML} -計算: モナド型を明示的に持つプログラミング言語 (Haskell)
E. Moggi. Notions of computation and monads. Information and Computation 93 (1), 1991

計算効果つき式代入と分岐を持つ命令形言語の意味論

言語 Imp_0

Var : 変数集合, $CExp$: 計算効果付き式の集合, $BExp$: 条件式の集合

$$P \triangleq I^* \quad I \triangleq x \leftarrow c \mid if(b)\{P\}else\{P\} \quad (x \in Var, b \in BExp, c \in CExp)$$

Imp_0 の解釈のために用意するデータ (以下 $Env = \mathbb{N}^{Var}$)

(M, η, kl)	圏 \mathbf{Set} 上のモナド	計算効果のモデル
$\llbracket c \rrbracket$	$Env \rightarrow M(\mathbb{N})$	計算効果付き式 $c \in CExp$ の意味
$\llbracket b \rrbracket$	$Env \rightarrow \{\text{true}, \text{false}\}$	条件式 $b \in BExp$ の意味

解釈の基本方針:

$$\llbracket P \rrbracket : Env \rightarrow M(Env), \quad \llbracket I \rrbracket : Env \rightarrow M(Env)$$

順次実行の解釈

どのように順次実行を解釈するか?

$$\frac{[\![P_1]\!] : Env \rightarrow M(Env) \quad [\![P_2]\!] : Env \rightarrow M(Env)}{[\![P_1; P_2]\!] = ??? : Env \rightarrow M(Env)}$$

モナドの $kl_{X,Y} : X \Rightarrow M(Y) \rightarrow M(X) \Rightarrow M(Y)$ を使う

$$\frac{[\![P_1]\!] : Env \rightarrow M(Env) \quad [\![P_2]\!] : Env \rightarrow M(Env)}{[\![P_1; P_2]\!] = kl([\![P_2]\!]) \circ [\![P_1]\!] : Env \rightarrow M(Env)}$$

例: $\Sigma = \{go_\sigma, buy_\sigma, eat_\sigma \mid \sigma : \text{文字列}\}, E = \emptyset$ とし $M = T_\Sigma^E$ の時

$$\begin{aligned} [\![P_1]\](\rho) &= go_{cafe}(eat_{cake}(\rho')) \\ [\![P_2]\](\rho') &= go_{market}(buy_{bread}(\rho'')) \\ \implies [\![P_1; P_2]\](\rho) &= go_{cafe}(eat_{cake}(go_{market}(buy_{bread}(\rho'')))) \\ &= sub([\![P_2]\]) \circ [\![P_1]\](\rho) \end{aligned}$$

$[\![x \leftarrow c]\]$ の解釈には後述するストレングスを用いる

Computational λ -calculus λ_c

Var: 変数集合, ar : Con $\rightarrow \mathbb{N}$: 計算効果定数の仕様

$\sigma, \tau ::= \text{nat} \mid \text{unit} \mid \tau * \tau \mid \tau \rightarrow \tau$

$M ::= x \mid () \mid (M, M) \mid (\pi_i M) \mid (\lambda x . M) \mid (MM) \mid c \mid \text{let } x \text{ be } M \text{ in } M$
($x \in \text{Var}, c \in \text{Con}, i \in \{1, 2\}$)

(注: [Moggi89] から計算型 T_τ を抜いている)

型付け規則は普通の単純型付ラムダ計算に以下を追加

$$\frac{c \in \text{Con}^{(n)}}{\Gamma \vdash c : \underbrace{\text{nat} * \cdots * \text{nat}}_n \rightarrow \text{nat}}$$

$$\frac{\Gamma \vdash M : \tau \quad \Gamma, x : \tau \vdash N : \sigma}{\Gamma \vdash \text{let } x \text{ be } M \text{ in } N : \sigma}$$

「 M を実行し、その返り値を x に割当てて N を実行」

値判定規則 $M \downarrow$ と等式理論 $\Gamma \vdash M = N : \tau$ を導入 (省略)

値呼び関数型プログラミング言語を単純化・理想化したものに対応

λ_c の集合論的解釈

解釈のために用意するデータ

(M, η, kl)	圏 Set 上のモナド	計算効果のモデル
$\llbracket c \rrbracket$	$\mathbb{N}^n \rightarrow M(\mathbb{N})$	計算効果定数 $c \in Con^{(n)}$ の意味

型の解釈

$$\begin{aligned}\llbracket \text{nat} \rrbracket_c &= \mathbb{N}, & \llbracket \text{unit} \rrbracket_c &= \{()\}, & \llbracket \tau_1 * \tau_2 \rrbracket_c &= \llbracket \tau_1 \rrbracket_c \times \llbracket \tau_2 \rrbracket_c, \\ \llbracket \tau \rightarrow \sigma \rrbracket_c &= \llbracket \tau \rrbracket_c \Rightarrow M(\llbracket \sigma \rrbracket_c)\end{aligned}$$

型判定の解釈の方針

$$\llbracket x_1 : \tau_1, \dots, x_n : \tau_n \vdash M : \tau \rrbracket_c : \prod \llbracket \tau_i \rrbracket_c \rightarrow M(\llbracket \tau \rrbracket_c)$$

ML, OCaml などでプログラムを動かすと計算効果が起こることに対応

組の解釈とストレングス

組の解釈

$$\frac{[\![M]\!]_c : [\![\Gamma]\!]_c \rightarrow \textcolor{blue}{M}([\![\tau_1]\!]_c) \quad [\![N]\!]_c : [\![\Gamma]\!]_c \rightarrow \textcolor{blue}{M}([\![\tau_2]\!]_c)}{[\![(M, N)]\!]_c = \textcolor{red}{???} : [\![\Gamma]\!]_c \rightarrow \textcolor{blue}{M}([\![\tau_1]\!]_c \times [\![\tau_2]\!]_c)}$$

問: $c_1 \in \textcolor{blue}{M}(X)$ と $c_2 \in \textcolor{blue}{M}(Y)$ から $\textcolor{blue}{M}(X \times Y)$ の元を作るには?

例: $c_1 = go_{\text{cafe}}(100) \in \textcolor{blue}{M}(\text{JPY})$, $c_2 = go_{\text{market}}(\text{チラシ}) \in \textcolor{blue}{M}(\text{Paper})$ の時

左優先 $go_{\text{cafe}}(go_{\text{market}}(100, \text{チラシ})) \in \textcolor{blue}{M}(\text{JPY} \times \text{Paper})$

右優先 $go_{\text{market}}(go_{\text{cafe}}(100, \text{チラシ})) \in \textcolor{blue}{M}(\text{JPY} \times \text{Paper})$

これらをどう作るか?

答: 圏 **Set** 上のモナドに備わる唯一のストレングス

$\theta_{X,Y} : X \times \textcolor{blue}{M}(Y) \rightarrow \textcolor{blue}{M}(X \times Y)$ を使う [Kock72, Moggi89]。

$$\theta_{X,Y}(x, c) \triangleq kl(\lambda y \in Y . \eta_{X \times Y}(x, y))$$

組の解釈とストレングス

ストレングス $\theta_{X,Y}$ から余ストレングス $\theta'_{X,Y} : \mathbf{M}(X) \times Y \rightarrow \mathbf{M}(X \times Y)$ を

$$\mathbf{M}(X) \times Y \xrightarrow{\text{sw}_{\mathbf{M}(X),Y}} Y \times \mathbf{M}(X) \xrightarrow{\theta_{Y,X}} \mathbf{M}(Y \times X) \xrightarrow{\text{kl}(\eta \circ \text{sw}_{Y,X})} \mathbf{M}(X \times Y)$$

の合成で定義 ($\text{sw}_{X,Y}(x,y) \triangleq (y,x)$)。これにより

左優先 $Lp \triangleq \text{kl}(\theta) \circ \theta' : \mathbf{M}(X) \times \mathbf{M}(Y) \rightarrow \mathbf{M}(X \times Y)$

右優先 $Rp \triangleq \text{kl}(\theta') \circ \theta : \mathbf{M}(X) \times \mathbf{M}(Y) \rightarrow \mathbf{M}(X \times Y)$

の二通りの関数を得る。これにより

左優先評価 $\llbracket (M,N) \rrbracket_c(\rho) = Lp(\llbracket M \rrbracket_c(\rho), \llbracket N \rrbracket_c(\rho))$

右優先評価 $\llbracket (M,N) \rrbracket_c(\rho) = Rp(\llbracket M \rrbracket_c(\rho), \llbracket N \rrbracket_c(\rho))$

$Lp = Rp$ となる圏 **Set** 上のモナド (M, η, kl) は可換であると言う。

- Maybe、(有限) 幕集合、(有限) 確率分布モナドは可換
- ライター、(有限) 状態、継続モナドは非可換

Computational metalanguge λ_{ML}

Var: 変数集合, ar : Con $\rightarrow \mathbb{N}$: 計算効果定数の仕様

$\sigma, \tau ::= \text{nat} \mid \text{unit} \mid \tau * \tau \mid \tau \rightarrow \tau \mid T\tau$

$M ::= \dots \text{(単純型付ラムダ計算と同じ)} \dots \mid c \mid [M] \mid \text{let } x \text{ be } M \text{ in } M$

型付け規則は普通の単純型付ラムダ計算に以下を追加

$$\frac{c \in \text{Con}^{(n)}}{\Gamma \vdash c : \underbrace{\text{nat} * \cdots * \text{nat}}_n \rightarrow T(\text{nat})}$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash [M] : T\tau} \text{ return} \quad \frac{\Gamma \vdash M : T\tau \quad \Gamma, x : \tau \vdash N : T\sigma}{\Gamma \vdash \text{let } x \text{ be } M \text{ in } N : T\sigma} \text{ let}$$

$=_{\beta\eta}$ を拡張した等式理論を導入

モナド型を組み込んだラムダ計算 (= Haskell のような言語)

解釈のために用意するデータ

(M, η, kl)	圏 Set 上のモナド	計算効果のモデル
$\llbracket c \rrbracket$	$\mathbb{N}^{ar(c)} \rightarrow \textcolor{blue}{M}(\mathbb{N})$	計算効果定数 $c \in Con^{(n)}$ の意味

型の解釈

$$\begin{aligned}\llbracket \text{nat} \rrbracket_m &= \mathbb{N}, & \llbracket \text{unit} \rrbracket_m &= \{()\}, & \llbracket \tau_1 * \tau_2 \rrbracket_m &= \llbracket \tau_1 \rrbracket_m \times \llbracket \tau_2 \rrbracket_m, \\ \llbracket \tau \rightarrow \sigma \rrbracket_m &= \llbracket \tau \rrbracket_m \Rightarrow \llbracket \sigma \rrbracket_m, & \llbracket T\tau \rrbracket_m &= \textcolor{blue}{M}(\llbracket \tau \rrbracket_m)\end{aligned}$$

型判定の解釈の方針

$$\llbracket x_1 : \tau_1, \dots, x_n : \tau_n \vdash M : \tau \rrbracket_m : \prod \llbracket \tau_i \rrbracket_m \rightarrow \llbracket \tau \rrbracket_m$$

let の所でストレンジスを使う

$$\begin{aligned}\llbracket [M] \rrbracket_m(\rho) &= \eta \circ \llbracket M \rrbracket_m(\rho) \\ \llbracket \text{let } x \text{ be } M \text{ in } N \rrbracket_m(\rho) &= kl(\llbracket N \rrbracket_m) \circ \theta(\rho, \llbracket M \rrbracket_m \rho)\end{aligned}$$

λ_c から λ_{ML} への変換 (左優先)

$$\Psi(\text{nat}) = \text{nat}, \quad \Psi(\text{unit}) = (), \quad \Psi(\tau * \sigma) = \Psi(\tau) * \Psi(\sigma)$$

$$\Psi(\tau \rightarrow \sigma) = \Psi(\tau) \rightarrow T\Psi(\sigma)$$

$$\Psi(x) = [x]$$

$$\Psi(()) = [()]$$

$$\Psi((M, N)) = \mathbf{let} \ x \ \mathbf{be} \ \Psi(M) \ \mathbf{in} \ \mathbf{let} \ y \ \mathbf{be} \ \Psi(N) \ \mathbf{in} \ [(x, y)]$$

$$\Psi(\pi_i M) = \mathbf{let} \ x \ \mathbf{be} \ \Psi(M) \ \mathbf{in} \ [\pi_i x]$$

$$\Psi(\lambda x . M) = [\lambda x . \Psi(M)]$$

$$\Psi(MN) = \mathbf{let} \ f \ \mathbf{be} \ \Psi(M) \ \mathbf{in} \ \mathbf{let} \ x \ \mathbf{be} \ \Psi(N) \ \mathbf{in} \ fx$$

$$\Psi(c) = c$$

$$\Psi(\mathbf{let} \ x \ \mathbf{be} \ M \ \mathbf{in} \ N) = \mathbf{let} \ x \ \mathbf{be} \ \Psi(M) \ \mathbf{in} \ \Psi(N)$$

定理

$$[\![\Gamma \vdash M : \tau]\!]_c = [\![\Psi(\Gamma) \vdash \Psi(M) : T\Psi(\tau)]!]_m \text{ (ただし } [\![-]\!]_c \text{ は左優先)}$$

Part IV

二つのモナド的意味論の比較

モナド的意味論の比較

0か1を選んで返す命令 $\text{choose} : 0$ を持つ λ_{ML} を考える。

$$M \triangleq \mathbf{let} \, x \, \mathbf{be} \, \text{choose}() \, \mathbf{in} \, \mathbf{let} \, y \, \mathbf{be} \, \text{choose}() \, \mathbf{in} \, [x + y]$$

様々なモナド的意味論が考えられる

モナド $M(X)$	$\llbracket \text{choose} \rrbracket() \in M(\mathbb{N})$	$\llbracket M \rrbracket_m$
リスト X^*	$(0, 1)$	$(0, 1, 1, 2)$
幂集合 $\{U \subseteq X\}$	$\{0, 1\}$	$\{0, 1, 2\}$
継続 $(X \Rightarrow R) \Rightarrow R$	$\lambda k . k(0) \vee k(1)$	$\lambda k . k(0) \vee k(1) \vee k(2)$

注: R は結び半束とする

以下は成立するか?

$$\llbracket M : T\text{nat} \rrbracket_m^{\text{幂集合}} = \{x \mid x \text{ は } \llbracket M : T\text{nat} \rrbracket_m^{\text{リスト}} \text{ に出現}\}$$

$$\llbracket M : T\text{nat} \rrbracket_m^{\text{継続}} = \lambda k . \bigvee \{k(x) \mid x \in \llbracket M : T\text{nat} \rrbracket_m^{\text{幂集合}}\}$$

計算効果比較問題

- モナド (M_i, η^i, k^i) と $\llbracket c \rrbracket^i : \mathbb{N}^n \rightarrow M_i(\mathbb{N})$ ($i = 1, 2$)
- 二つの計算効果のモデルの間に期待する $C_{\text{nat}} \subseteq M_1(\mathbb{N}) \times M_2(\mathbb{N})$

問: 計算効果シミュレーション問題

任意の $x_1 : \text{nat}, \dots, x_n : \text{nat} \vdash M : T_{\text{nat}}$ に対し以下は成立するか?

$$\forall \rho \in \mathbb{N}^n . (\llbracket M \rrbracket_m^1(\rho), \llbracket M \rrbracket_m^2(\rho)) \in C_{\text{nat}}$$

定理

[K: Relating Computational Effects by T T-Lifting. ICALP (2) 2011: 174-185]

以下が成立すれば上の問題の答えは Yes

- ① 任意の $c \in Con^{(n)}, \rho \in \mathbb{N}^n, f_1 : \mathbb{N} \rightarrow M_1(\mathbb{N}), f_2 : \mathbb{N} \rightarrow M_2(\mathbb{N})$ に対し

$$(\forall x \in \mathbb{N} . (f_1(x), f_2(x)) \in C_{\text{nat}})$$

$$\implies (k^1(f_1)(\llbracket c \rrbracket^1(\rho)), k^2(f_2)(\llbracket c \rrbracket^2(\rho))) \in C_{\text{nat}}$$

- ② 任意の $x \in \mathbb{N}$ に対し $(\eta_{\mathbb{N}}^1(x), \eta_{\mathbb{N}}^2(x)) \in C_{\text{nat}}$

計算効果比較問題

証明の方針:

- ① 型に関する帰納法で二項関係を構成

$$R_{\tau} \subseteq \llbracket \tau \rrbracket_m^1 \times \llbracket \tau \rrbracket_m^2 \quad (R_{\text{nat}} = \{(n, n) \mid n \in \mathbb{N}\})$$

- ② 論理関係の基本補題: 任意の $\Gamma \vdash M : \tau$ に対し、

$$\forall (\rho_1, \rho_2) \in R\Gamma . (\llbracket M \rrbracket_m^1(\rho_1), \llbracket M \rrbracket_m^2(\rho_2)) \in R_{\tau}$$

- ③ $R(T_{\text{nat}}) \subseteq C_{\text{nat}}$ を示す

一般のモナドの組 M_1, M_2 に対し、3 を満たす **モナド型のための論理関係**をどのように作るか?

意味論的 TT-持ち上げ [K05]

λ_{ML} の二つの解釈の間の論理関係 $R(\tau) \subseteq \llbracket \tau \rrbracket_m^1 \times \llbracket \tau \rrbracket_m^2$ を以下で定義

$$R(\text{nat}) \triangleq \{(n, n) \mid n \in \mathbb{N}\} \quad R(\text{unit}) \triangleq \{\text{(()), ()}\} \quad \dots \quad R(T\tau) \triangleq (R(\tau))^{\top\top}$$

ここで $(-)^\top : P(X \times Y) \rightarrow P(\textcolor{blue}{M}_1(X) \times \textcolor{blue}{M}_2(Y))$ は意味論的 TT-持ち上げ

$$A^\top \triangleq \{(f_1, f_2) \mid \forall (x_1, x_2) \in A . (f_1(x_1), f_2(x_2)) \in C_{\text{nat}}\}$$

$$A^{\top\top} \triangleq \{(c_1, c_2) \mid \forall (f_1, f_2) \in A^\top . (kl^1(f_1)(c_1), kl^2(f_2)(c_2)) \in C_{\text{nat}}\}$$

これは [Lindley&Stark05] の leapfrog method の意味論化

命題

論理関係の基本補題が成立 (条件 1 を利用)

命題

$$R(T\text{nat}) = (R(\text{nat}))^{\top\top} \subseteq C_{\text{nat}} \text{ (条件 2 を利用)}$$

Sam Lindley, Ian Stark: Reducibility and TT-Lifting for Computation Types. TLCA 2005: 262-277

S. Katsumata: A Semantic Formulation of TT-Lifting and Logical Predicates for Computational Metalanguage. CSL 2005: 87-102

まとめ

- 圈 **Set** 上のモナドを代数的構造から導入
- 計算効果を持つプログラミング言語の意味に利用
3種類: 命令形、値呼び関数型、モナド型を持つ関数型
ストレングス $\theta : X \times M(Y) \rightarrow M(X \times Y)$ を利用
- 二つの λ_{ML} のモナド的意味論の比較
モナド型の論理関係に意味論的 $\top\top$ -持ち上げを用いる

発展的な話題

- 一般の圏における計算効果のモデル
ストレングス付きモナドを用いる
- call-by-push-value[Levy99]: call-by-name と call-by-value の統合
- モナドの一般化
パラメタライズドモナド [Atkey06]、次数付きモナド [K14]
- ハンドラー [Plotkin&Pretnar14]: 例外処理の一般化

Paul Blain Levy: Call-by-Push-Value: A Subsuming Paradigm. TLCA 1999: 228-242

Robert Atkey: Parameterised Notions of Computation. MSFP@MPC 2006

Shin-ya Katsumata: Parametric effect monads and semantics of effect systems. POPL 2014: 633-646

Gordon D. Plotkin, Matija Pretnar: Handlers of Algebraic Effects. ESOP 2009: 80-94