数学入門公開講座

平成８年８月５日（月）から８月９日（金）まで

京都大学数理解析研究所

講師及び内容

1．プログラミング言語、状態と型 （6時間15分）

京都大学数理解析研究所・助手　ガリグ　ジャック

チューリング機械とラムダ計算（帰納関数論）という、最も一般的な二つのコンピュータの抽象化が根本的に違う点は状態の扱いである。前者では、無限な書き込み可能なテープという形で残されているのに、後者では跡形もなく消えている。プログラミング言語を研究するには、後者が望ましいので、状態を再び導入し、ラムダ計算が持っている型（範疇）体系に埋め込む必要がある。

本講座では、両方の抽象化を紹介し、状態に関する諸問題を考える。

2．リーマン面 （6時間15分）

京都大学数理解析研究所・助教授　古田　幹雄

一変数の多項式 f (z) は、複素数 z に対して別の複素数 f (z) を対応させる写像です。z は複素数全体のなす平面の上を動きます。ここで、平面のかわりに球面や、トーラスを考えたらどんな世界が広がっているでしょうか。リーマン・ロッホの定理の紹介を目標とします。

3．「漸近挙動を巡って：太鼓の形と酔歩」（6時間15分）

京都大学数理解析研究所・教授　髙橋陽一郎

“Can you hear the shape of a drum?” は故 M. Kac の有名な論文のタイトルである。鼓や太鼓やドラムの音を聴いて、その幾何学的な形がわかるかという問いかけである。

また、酔歩は乱歩（random walk）とも呼ばれ，デタラメさ（randomness）とは何かを数学として問うときに最も基本的な、現代確率論の概念である。

これらについて語ることを通じて、漸近挙動ということばの意味をわかってもらえることを期待しつつ、解析学における「等式」の意味を考えてみたい。

時　間　割

日 / 時　間	8月5日(月)	6日(火)	7日(水)	8日(木)	9日(金)
10：30～11：45	ガリグ	ガリグ	ガリグ	ガリグ	ガリグ
11：45～13：00	休　憩				
13：00～14：15	古田	古田	古田	古田	古田
14：15～14：45	休　憩				
14：45～16：00	髙橋	髙橋	髙橋	髙橋	髙橋

平成8年度(第18回)数学入門公開講座テキスト(京都大学数理解析研究所, 平成8年8月5日～8月9日開催)

プログラミング言語、状態と型

京都大学数理解析研究所・助　手　ガリグ ジャック

1996, AUGUST 5,6,7,8,9, 10:30 ~ 11:45

プログラミング言語、状態と型

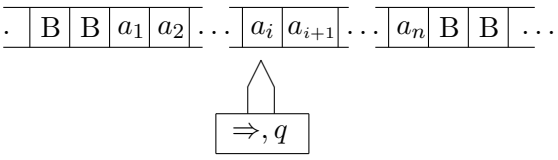
Jacques Garrigue

1996 年 8 月

この講義ノートの構成は次の通り。まず、第 1 章ではチューリング機械を定義し、いくつかの例をみる。第 2 章で型なしラムダ計算の定義と性質をみ、第 3 章で前者と後者の同値性を証明する。さらに第 4 章でラムダ計算に型体系を導入する。

1　チューリング機械

チューリング機械とは、計算機の全ての機能を備える最も単純なモデルである。英国の数学者 A. M. Turing が 1936 年に計算過程の定式化のために定義した。**記号**が書かれている**テープ**、それを読む**ヘッド**と制御部の**状態**という三つ組からできている。テープの長さは無限だが、記号のアルファベットと制御部の状態数は有限である、さらにテープの上に書かれている記号の数も各時点では有限である。



直感的なチューリング機械の動作は三段階になっている。

1. ヘッドの下のテープの値 a_i を読む。
2. その値と制御部の状態 q に応じて、新しい値 a_i' を同じ場所に書き込み、新しい状態 q' に移る。
3. 同様に次の移動方向が決り、書き終わった後、左または右の位置 ($i-1$ または $i+1$) にヘッドを移す。

このとても単純な機械では、実はコンピュータができる全ての計算が実行できる。
チューリング機械の使う記号、取れる状態、とその状態遷移を正確に定義する。

定義 1 チューリング機械は次の 5 つ組 $M = (K, \Sigma, \delta, q_0, H)$ によって定義される。

K: 空でない有限集合。K の要素を**状態**という。

Σ: 空でない有限集合 (アルファベット)。Σ の元を**記号**という。Σ は**空白記号** B を含む。

q_0: K の要素で、**初期状態**という。

H: K の部分集合で、その要素を**停止状態**という。

δ: $(K \setminus H) \times \Sigma \to \Sigma \times \{ \text{左}, \text{右} \} \times K$ なる関数で、**遷移関数**という。

それでは機械が定義されたが、動的な状態はまだ把握されていない。そのために**時点**の概念が導入される。

定義 2 あるチューリング機械 M の**時点**はテープ、位置、状態の 3 つ組で定義される。

$$t = (T, i, q)$$

T: $\mathbf{Z} \to K$ なる関数で、$T(n) \neq \mathrm{B}$ であるような n は高々有限個しかない。

i: $\mathbf{Z}$ の整数。

q: K の状態。

定義 3 チューリング機械 M が一動作で時点 t から t' に移ることを $t \vdash_M t'$ と書く。

$$\delta(q, T(i)) = (a, d, q') \Rightarrow (T, i, q) \vdash_M (T', i', q')$$

- $T'(i) = a$, $k \neq i$ ならば $T'(k) = T(k)$
- $d =$ 右 ならば $i' = i + 1$, $d =$ 左 ならば $i' = i - 1$

$\vdash_M$ の反射推移閉包を $\vdash_M^*$ と書く。さらに、q' は停止状態ならば、$(T, 0, q_0) \vdash_M^* (T', n, q')$ を $T \triangleright_M (T', n, q')$ と書き、M を T で実行した結果が (T', n, q') だという。

例題 1 括弧の対応をチェックする機械。

Σ = $\{\mathrm{B}, \mathrm{E}, \mathrm{L}, \mathrm{R}\}$
B は空記号、E は消した跡、L は左括弧、R は右括弧

K = $\{R, L, C, T, F\}$
R は右探索、L は左探索、C は最終チェック、T, F は真と偽

q_0 = R

H = $\{T, F\}$

δ =

$q \backslash a$	B	E	L	R
R	$(a, 左, C)$	$(a, 右, R)$	$(a, 右, R)$	$(\mathrm{E}, 左, L)$
L	$(a, 右, F)$	$(a, 左, L)$	$(\mathrm{E}, 右, R)$	$(a, 右, F)$
C	$(a, 右, T)$	$(a, 左, C)$	$(a, 右, F)$	$(a, 右, F)$

例題 2 任意の自然数の加算 $a + b = c$

テープの初期状態 (それ以外は B). $a = \sum_0^k a_i \cdot 2^i, b = \sum_0^l b_i \cdot 2^j$

a_0	a_1	$\ldots$	a_k	M	I_0	b_0	$\ldots$	b_l

テープの最終状態 (それ以外は B). $c = \sum_0^m c_i \cdot 2^i$

M	c_0	c_1	$\ldots$	c_m

$$
\begin{aligned}
\Sigma &= \{\mathrm{B}, 0, 1, \mathrm{M}, \mathrm{I}_0, \mathrm{I}_1\} \\
K &= \{L, R, A_0, A_1, A_0', A_1', A_0'', A_2'', W_0, W_1, F, T\} \\
q_0 &= R
\end{aligned}
$$

δ =

$q\backslash a$	B	0	1	M	I_0	I_1
L	$(a, 右, R)$	$(a, 左, L)$	$(a, 左, L)$	$(a, 左, L)$		
R		$(\mathrm{B}, 右, A_0)$	$(\mathrm{B}, 右, A_1)$	$(a, 右, F)$		
A_i	$(i, 左, L)$	$(a, 右, A_i)$	$(a, 右, A_i)$	$(a, 右, A_i)$	$(a, 右, A_i')$	$(a, 右, A_i'')$
A_i'	$(\mathrm{I}_0, 左, W_i)$	$(\mathrm{I}_0, 左, W_i)$	$(\mathrm{I}_i, 左, W_{1-i})$			
A_i''	$(\mathrm{I}_i, 左, W_{1-i})$	$(\mathrm{I}_i, 左, W_{1-i})$	$(\mathrm{I}_1, 左, W_i)$			
W_i					$(i, 左, L)$	$(i, 左, L)$
F	$(a, 左, T)$	$(a, 右, F)$	$(a, 右, F)$		$(a, 右, A_0')$	$(a, 右, A_0'')$

2 ラムダ計算

ドイツの論理学者 Schonfinkel が 1920 年代に作ったλ計算は元々論理の証明論のための道具だった。しかし、情報科学との関係がだんだん深くなり、1950 年代からリスプから始まる関数型言語の重要な基礎となった。ラムダ計算自体が機械だとはいえないが、それを用いて様々な計算機械がすぐ導かれる。

2.1 (抽象) 項書換系

元々はそうではなかったが、λ計算は項書換系によって定義される。計算を項の一部の書き換えとして見る、ということだ。例えば、数式の書換系を次のように定義できる。

項 (term) $E ::= \mathcal{R} \mid (E + E) \mid (E - E) \mid (E \times E) \mid (E/E)$

書換規則 x, y は $\mathcal{R}$ の実数ならば、

$$
\begin{aligned}
(x + y) &\to x + y \\
(x - y) &\to x - y \\
(x \times y) &\to x \times y \\
(x/y) &\to x/y
\end{aligned}
$$

($(x + y)$ は数式だが、 $x + y$ は実数である)

上の規則を簡約規則ともいう。

簡約の例

$$(15 + (1/3)) \times (5 - 2) \to (15 + 0.3333) \times (5 - 2) \to (15.3333) \times 3 \to 46$$

2.2 λ計算の構文 (syntax)

定義 4 λ項 (λ-term) は次の三つの構文からできている。

$$\begin{array}{rcll} M & ::= & x & \text{変数 (variable)} \\ & | & \lambda x.M & \text{抽象 (abstraction)} \\ & | & (M\ M) & \text{適用 (application)} \end{array}$$

x は文脈 (環境) で定義されている値 (別のλ項) を指す。

$\lambda x.M$ は項 M の中に使われている変数 x を束縛する。関数として見れば $f = \lambda x.M$ という定義は、普段の $f(x) = M$。ただし、$\lambda x.M$ と書くとき、名前をつける必要がないという便利さがある。

$(M_1\ M_2)$ は関数の適用である。普段の $M_1(M_2)$ にあたるが、M_1 は名前 (変数) だけでなく、どのλ項でもよい。

上の文法を普段の数式と混ぜると、

$$f(x) = x + 1 \text{ のとき } f(2)$$

が 1 回で書ける。

$$((\lambda x.x + 1)\ 2)$$

自由変数と代入 $\lambda x.M$ では、M の中の全ての x の出現は「束縛されている」という。ある項 M の中で、x が使われながら、束縛されていないのであれば、x は M の自由変数である。M の自由変数の集合 $FV(M)$ は再起的に次の三条で定義される。

$$\begin{array}{lcl} FV(x) & = & \{x\} \\ FV(\lambda x.M) & = & FV(M) \setminus \{x\} \\ FV(M\ N) & = & FV(M) \cup FV(N) \end{array}$$

代入はそういう自由変数の値を定義する。$([N/x]M)$ は項 M の中に出現するすべての自由な x を N と置き換える。

$$\begin{array}{lcl} ([N/x]x) & = & N \\ ([N/x]y) & = & y \\ ([N/x]\lambda x.M) & = & \lambda x.M \\ ([N/x]\lambda y.M) & = & \lambda y.([N/x]M) \\ ([N/x](M\ M')) & = & (([N/x]M)\ ([N/x]M')) \end{array}$$

4 条目では、y は N の自由変数であってはならない。そのためには、次のα変換が許される。z は M の自由変数でなければ、

$$\lambda y.M \leftrightarrow \lambda z.([z/y]M)$$

こういう束縛変数の改称は常に許される。

2.3 簡約規則

定義 5 λ項、α変換と次のβ簡約で構成される項書き換え系はλ計算という。

$$((\lambda x.M)\ N) \to ([N/x]M)$$

例題 3 簡約の例

$$
\begin{array}{ll}
 & (\lambda f.\lambda g.\lambda x.f\ x\ (g\ x))\ (\lambda x.\lambda y.x)\ (\lambda x.\lambda y.x) \\
\to & \lambda x.((\lambda x.\lambda y.x)\ x\ ((\lambda x.\lambda y.x)\ x)) \\
\to & \lambda x.((\lambda y.x)\ (\lambda y.x)) \\
\to & \lambda x.x
\end{array}
$$

$$
\begin{array}{ll}
 & (\lambda x.(x\ x))\ (\lambda x.(x\ x)) \\
\to & (\lambda x.(x\ x))\ (\lambda x.(x\ x)) \\
\to & \ldots
\end{array}
$$

定理 1 λ計算は合流性を持つ。$M \to \ldots \to N$ と $M \to \ldots \to P$ という二つの簡約があれば、$N \to \ldots \to T$, $P \to \ldots \to T$ となるような T が存在する。

2.4 λ計算は万能である

全てのプログラムはλ計算で書ける。

整数 チャーチによるエンコーディングがある (Church numeral)。

$$
\begin{array}{ll}
\mathsf{c}_n = \lambda f.\lambda x.(f\ \ldots(f\ x)\ldots) & f \text{ を } n \text{ 回適用する} \\
\mathsf{c}_+ = \lambda m.\lambda n.\lambda f.\lambda x.(m\ f\ (n\ f\ x)) & \text{加算}
\end{array}
$$

ブール環 次のエンコーディングを使う。

$$
\mathsf{t} = \lambda x.\lambda y.x \qquad \mathsf{f} = \lambda x.\lambda y.y \qquad \mathsf{not} = \lambda b.\lambda x.\lambda y.(b\ y\ x)
$$

数の区別も可能で、例えば次の項が引き数に数をもらい、真偽値を返す。

$$
\mathsf{if0} = \lambda n.(n\ (\lambda x.\mathsf{f})\ \mathsf{t})
$$

不動点演算子 再起的な関数を定義するのに、不動点演算子 Y が必要になる。その基本的な属性は、$(Y\ M)$ が $(M\ (Y\ M))$ に簡約できることである。

$$
Y = (\lambda f.\lambda x.(x\ (f\ f\ x)))\ (\lambda f.\lambda x.(x\ (f\ f\ x)))
$$

それによって、引き算なども定義できる。

$$
\begin{array}{lcl}
\mathsf{c}_- & = & (Y\ (\lambda f.\lambda x.\lambda y.(\mathsf{if0}\ y\ x\ (f\ (\mathsf{p}\ x)\ (\mathsf{p}\ y)))))) \\
\mathsf{s} & = & \lambda n.\lambda f.\lambda x.(f\ (n\ f\ x)) \\
\mathsf{p} & = & \lambda n.(n\ (\lambda z.(\mathsf{if0}\ z\ \mathsf{f}\ (\mathsf{s}\ z)))\ \mathsf{f})
\end{array}
$$

2.5 評価戦略

ラムダ計算は機械ではない。なぜかといえば、簡約という計算機構があっても、それをどの順番で行うか、あるいはいつ計算が終わるか、が定まっていない。標準形と戦略はちょうどそれを決める役割をはたしている。

標準形　簡約できる個所を全く残さない (既約) 標準形以外にも、先頭の部分だけが簡約できない弱冠頭標準形も定義できる。例えば $(x\ ((\lambda y.y)\ z))$ や $\lambda x.((\lambda y.y)\ z)$ は既約標準形でない (y がまだ簡約できる) が弱冠頭標準形である。形式化すると、$\lambda x.M$ または $(x\ M_1\ \ldots\ M_n)$ のどちらかの形をしたものが弱冠頭標準形である。

最左戦略　最も左にある簡約できる個所を先に簡約する。名前呼び出し (call-by-name) に当る。関数の引数を評価せずに、そのまま代入する。$(\lambda x.x)\ ((\lambda y.y)\ z) \to ((\lambda y.y)\ z)$。ある戦略で $M \to^* N(N$ 標準形) のとき、最左戦略でも $M \to^* N$。

最右最内戦略　最も右にある簡約できる個所の中で、最も中にあるものを選ぶ。値呼び出し (call-by-value) に当る。関数の引数を標準形に落してから代入する。$(\lambda x.x)\ ((\lambda y.y)\ z) \to ((\lambda x.x)\ z)$。ある戦略で $M \to^* \ldots$(無限に続く簡約) のとき、最右最内戦略でも無限な簡約になる。

機械　標準形 (停止状態) と戦略 (遷移関数) の組み合わせを選ぶと、任意のλ項を決定的な機械としてみることができる。

3　チューリング機械とラムダ計算の同値性

3.1　チューリング機械をλ項に

対とリスト構造　二つの値を一つの項で表現する時、対という形にする。$(a_1, a_2) = \lambda f.(f\ a_1\ a_2)$。適当な関数を使うと、各要素が抽出できる。例えば $(a_1, a_2)(\lambda x.\lambda y.x) = a_1$。便宜のために $fst = \lambda p.(p\ (\lambda x.\lambda y.x))$ と $snd = \lambda p.(p\ (\lambda x.\lambda y.y))$ を定義する。同様に n 組が作れる。$(a_1, \ldots a_n) = \lambda f.(f\ a_1\ \ldots\ a_n)$.

規則性のある無限なデータ構造を無限リストとして表現できる。$[a_1, a_2, \ldots]$ は $(a_1, [a_2, a_3, \ldots])$ $= \lambda f.(f\ a_1\ [a_2, a_3, \ldots_n])$ になる。このままでは無限なλ項になり、計算に使えないが、規則性があれば Y を使って有限なλ項が作れる。

状態と記号　$M = (K, \Sigma, q_0, H, \delta)$, $|K| = k$, $|\Sigma| = l$ とする。各状態と記号に番号をふっておき、$K = \{q_0, \ldots, q_{k-1}\}$, $\Sigma = \{\sigma_0 = \mathrm{B}, \ldots, \sigma_{l-1}\}$。次の翻訳を定義する。

$$\overline{q_i} = \lambda x_0 \ldots x_{k-1}.x_i$$
$$\overline{\sigma_i} = \lambda x_0 \ldots x_{l-1}.x_i$$

時点　時点 (T, n, q) を

$$(\overline{q}, \overline{T(n)}, [\overline{T(n-1)}, \overline{T(n-2)}, \ldots], [\overline{T(n+1)}, \overline{T(n+2)}, \ldots])$$

の 4 つ組で表す。無限リストを使うことになるが、B でしか構成されていない末端の部分を Y を使って表現する。$[\overline{\mathrm{B}}, \ldots] = Y(\lambda y.(\overline{\mathrm{B}}, y)) \to^* (\overline{\mathrm{B}}, Y(\lambda y.(\overline{\mathrm{B}}, y))) \to^* \ldots$

遷移関数　ある時点から次の時点への遷移関数を関数の l 組の k 組で表す。

$$\Delta = \lambda t.t((\overline{\delta}(q_0, \sigma_0), \overline{\delta}(q_0, \sigma_1), \ldots, \overline{\delta}(q_0, \sigma_{l-1})), \ldots, (\overline{\delta}(q_{k-1}, \sigma_0), \ldots))$$

$$\overline{\delta}(q,\sigma) = \begin{cases} \lambda t_l.\lambda t_r.f(\overline{q'}, fst\ t_l, snd\ t_l, (\overline{\sigma'}, t_r)) & \text{when } \delta(q,\sigma) = (\sigma', \text{左}, q') \\ \lambda t_l.\lambda t_r.f(\overline{q'}, fst\ t_r, (\overline{\sigma'}, t_l), snd\ t_r) & \text{when } \delta(q,\sigma) = (\sigma', \text{右}, q') \\ \lambda t_l.\lambda t_r.(\overline{q}, \overline{\sigma}, t_l, t_r) & \text{when } q \in H \end{cases}$$

この中の f を恒等関数 $\lambda x.x$ と定めると、抽象的に見る Δ の型は $(K, \Sigma, \Sigma\ list, \Sigma\ list) \to (K, \Sigma, \Sigma\ list, \Sigma\ list)$ であり、時点から時点への遷移関数に対応している。

実行　Δ は 1 回分の動作しか行わないので、その不動点を取ればよい。そのために f を抽象化する。

$$\lambda f.\Delta : \ ((K, \Sigma, \Sigma\ list, \Sigma\ list) \to (K, \Sigma, \Sigma\ list, \Sigma\ list)) \to ((K, \Sigma, \Sigma\ list, \Sigma\ list) \to (K, \Sigma, \Sigma\ list, \Sigma\ list))$$

$$(Y\ (\lambda f.\Delta)) : (K, \Sigma, \Sigma\ list, \Sigma\ list) \to (K, \Sigma, \Sigma\ list, \Sigma\ list)$$

それを初期状態に適用すると、$T \triangleright (T', n, q')$ が算出される。しかし無限リストを使っているので、既約標準形まで計算すると止らない。弱冠頭標準形では正しい結果が得られる。

$$(Y\ (\lambda f.\Delta))\ (\overline{q_0}, \overline{T(0)}, [\overline{T(-1)}, \ldots], [\overline{T(1)}, \ldots]) \to^* (\overline{q'}, \overline{T'(n)}, [\overline{T'(n-1)}, \ldots], [\overline{T'(n+1)}, \ldots])$$

3.2　λ項を実行できるチューリング機械

この節の目的はλ項をテープに書き、実行することである。

テープを実行するチューリング機械を考える前に、λ項を実行しやすい形でテープに書き込まなければならない。特に変数名とα変換は省きたい。そのために De Bruijn 添数を導入する。

定義 6 De Bruijn 添数　変数名を使わないラムダ計算を定義する。

$$M ::= n \mid \lambda M \mid (M\ M)$$

添数はこの変数が何番目の抽象 *(*中から数えて*)* で束縛されたかを表す。例えば $\lambda x.\lambda y.x = \lambda\lambda 2$, $\lambda x.(x\ x) = \lambda(1\ 1)$。代入を新しく定義する。

$$\begin{array}{llll} \Uparrow_n k = k+1 & k \geq n & [N/n]n = N & \\ \Uparrow_n k = k & k < n & [N/n]k = k-1 & k > n \\ \Uparrow_n \lambda M = \lambda(\Uparrow_{n+1} M) & & [N/n]k = k & k < n \\ & & [N/n]\lambda M = \lambda([\Uparrow_n N/n+1]M) & \\ \beta\text{簡約} \quad \lambda M\ N \to [N/1]M & & [N/n](M\ M') = ([N/n]M\ [N/n]M') & \end{array}$$

例題 4 対から 1 つ目の値を出す。

$$fst\ \lambda(1\ a\ b) = \lambda(1\ \lambda\lambda 2)\ \lambda(1\ a\ b) \to \lambda(1\ a\ b)\ \lambda\lambda 2 \to \lambda\lambda 2\ a\ b \to \lambda a\ b \to a$$

テープの形式　5 つの記号 $\{0, 1, \mathrm{E}, \lambda, @\}$ を使い、テープへの変換 $\overline{M}$ を定義する。

$$\begin{array}{rcl} \overline{n} & = & (n \text{ の 2 進法記述})\ \mathrm{E} \\ \overline{\lambda M} & = & \lambda\ \overline{M} \\ \overline{(M\ M')} & = & @\ \overline{M}\ \overline{M'} \end{array}$$

チューリング機械 Λ　上のテープを実行する機械 Λ はここで定義しないが、加算等の簡単な動作でできるので、そういうチューリング機械の存在は明かであろう。ここで弱冠頭標準形を最左戦略で計算する。

3.3　万能チューリング機械

任意のチューリング機械をλ項で表現できる、任意のλ項はある特定のチューリング機械が実行するテープに変換できる。そういう二つの定理に面白い系がある。

系 1 任意のチューリング機械を模擬できる万能チューリング機械が存在する。その機械は模擬する機械の定義をテープで与えられ、それを実行する。
証明　模擬したいチューリング機械をλ項に変換し、それを Λ に与えればよい。

4　型付きλ計算

上の関数は数学的関数と違い、定義域と値域を定めていない。例えば、c_+ は整数以外の物に適用しても意味がないが、そのような項は正しくないと判断できない。

4.1　型 (type) と項

型付きλ計算では、そのような領域を**型**で表す。単純な計算系では全ての値は一つだけの型に属する。型には二種類があり、基底型と関数・構造型に分けられる。

$$\begin{array}{rcl} b & ::= & \mathsf{int} \mid \mathsf{bool} \mid \ldots \\ t & ::= & b \mid t \to t \mid t \times t \end{array}$$

項の中にも、型情報を入れ、定数も追加する。

$$M ::= x \mid c_t \mid \lambda x{:}t.M \mid (M\ M) \mid (M, M)$$

簡約規則には、β に加えて定数に関する δ 規則も導入される。

$$\begin{array}{lcl} (\lambda x{:}\tau.M)\ N & \to & [N/x]M \\ (\mathsf{fst}_{\tau\times\theta\to\tau}\ (M,N)) & \to & M \\ (\mathsf{snd}_{\tau\times\theta\to\theta}\ (M,N)) & \to & N \\ (\mathsf{add}_{\mathsf{int}\to\mathsf{int}\to\mathsf{int}}\ m_{\mathsf{int}}\ n_{\mathsf{int}}) & \to & (m+n)_{\mathsf{int}} \\ \ldots & & \end{array}$$

4.2　型推論

ある項の正しさ (well-formedness) を型判定式で表す。

$$\Gamma \vdash M : \tau$$

M は項であり、τ は型である。そして Γ は型宣言列と呼ばれる変数名と型の集合 $(x_1 : \tau_1, \ldots, x_n : \tau_n)$。

定義 7 次の型推論規則で型判定式が証明された項を正しいとする。

$$\text{変数}\quad \Gamma \vdash x : \tau \quad (x : \tau \text{は} \Gamma \text{に含まれる})$$

$$\text{定数}\quad \Gamma \vdash c_\tau : \tau$$

$$\text{抽象}\quad \frac{\Gamma, x : \theta \vdash M : \tau}{\Gamma \vdash \lambda x{:}\theta.M : \tau}$$

$$\text{適用}\quad \frac{\Gamma \vdash M : \theta \to \tau \quad \Gamma \vdash N : \theta}{\Gamma \vdash (M\ N) : \tau}$$

$$\text{直積}\quad \frac{\Gamma \vdash M : \tau \quad \Gamma \vdash N : \theta}{\Gamma \vdash (M, N) : \tau \times \theta}$$

例題 5　証明

$$\dfrac{\dfrac{\dfrac{x : \mathsf{int} \vdash \mathsf{s}_{\mathsf{int}\to\mathsf{int}} : \mathsf{int} \to \mathsf{int} \quad x : \mathsf{int} \vdash x : \mathsf{int}}{x : \mathsf{int} \vdash (\mathsf{s}_{\mathsf{int}\to\mathsf{int}}\ x) : \mathsf{int}}}{\vdash \lambda x{:}\mathsf{int}.(\mathsf{s}_{\mathsf{int}\to\mathsf{int}}\ x) : \mathsf{int} \to \mathsf{int}} \qquad \vdash 1_{\mathsf{int}} : \mathsf{int}}{\vdash ((\lambda x{:}\mathsf{int}.(\mathsf{s}_{\mathsf{int}\to\mathsf{int}}\ x))\ 1_{\mathsf{int}}) : \mathsf{int}}$$

4.3　属性

次の定理は fst, snd 以外の δ 規則を含まない計算系に関するものである。

定理 2 (主部簡約) $\Gamma \vdash M : \tau$ と $M \to N$ が成り立てば、$\Gamma \vdash N : \tau$ が成り立つ。

定理 3 (強正規化) $\Gamma \vdash M : \tau$ ならば、無限な簡約列 $(M \to M_1 \to M_2 \to \ldots)$ は存在しない。

4.4　状態の導入

値呼び出し戦略　各関数がどの順番で評価されるかが分かるので、状態をどこでも変えられるようにしてもいい。動的に状態の大きさを増やしてもいい。

$$\begin{array}{lcll} \mathsf{newref}_{\tau\to\mathsf{ref}(\tau)}\ M & \to & r & \text{空いている位置 } r \text{ に値 } M \text{ を書き込む} \\ \mathsf{deref}_{\mathsf{ref}(\tau)\to\tau}\ r & \to & M & \text{位置 } r \text{ から値 } M \text{ を読む} \\ \mathsf{setref}_{\mathsf{ref}(\tau)\to\tau\to\mathsf{unit}}\ r\ M & \to & () & \text{位置 } r \text{ に値 } M \text{ を書き込む} \end{array}$$

例　$f(n)$ を呼ぶと、r の中身に n が足され、新しい値が返される。

$$r : \mathsf{ref}(\mathsf{int}) = \mathsf{newref}\ 0$$
$$f : \mathsf{int} \to \mathsf{int} = \lambda x.((\lambda y.((\lambda z.y)\ (\mathsf{setref}\ r\ y)))\ (\mathsf{add}\ (\mathsf{deref}\ r)\ x))$$

特定の評価戦略を仮定しない場合　状態を引数としてとり、結果と一緒に返す。もともと $f : \theta \to \tau$ なる関数があり、その関数の中で状態を変える必要があれば、実際には $f : \theta \to \sigma \to (\tau \times \sigma)$ という型になる (σ は状態全体の型)。このままでは使いにくいので、λ 計算を拡張する。

まず $\sigma \to (\tau \times \sigma) = ST(\tau)$ と略する。そして、この $ST(\tau)$ を扱う関数を定義する。ここでは $\sigma = \sigma_1 \times \ldots \times \sigma_n$

$$
\begin{array}{lcl}
\mathsf{get}i_{ST(\sigma_i)}\ (s_1,\ldots,s_n) & \to & (s_i,(s_1,\ldots,s_n)) \\
\mathsf{set}i_{\sigma_i\to ST(\mathsf{unit})}\ M\ (s_1,\ldots,s_n) & \to & ((),(s_1,\ldots,s_{i-1},x,s_{i+1},\ldots,s_n)) \\
\mathsf{then}_{ST(\theta)\to(\theta\to ST(\tau))\to ST(\tau)}\ M\ N\ s & \to & (\lambda p.(N\ (\mathsf{fst}\ p)\ (\mathsf{snd}\ p)))\ (M\ s) \\
\mathsf{return}_{\tau\to ST(\tau)}\ M\ s & \to & (M,s)
\end{array}
$$

この方法は圏論のモナドによるものである。

例 同上

$$
\begin{array}{l}
s : \mathsf{int}\times\mathsf{bool}\times\mathsf{int} = (0,\mathsf{t},0) \\
f : \mathsf{int}\to ST(\mathsf{int}) = \lambda x.(\mathsf{then}\ \mathsf{get3}\ (\lambda y.\mathsf{then}\ (\mathsf{set3}\ (\mathsf{add}\ x\ y))\ (\lambda z.\mathsf{get3})))
\end{array}
$$

参考文献

[1] 有川節夫. オートマトンと計算可能性, 情報処理シリーズ, 第 9 巻. 培風館, 1986.

[2] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, Vol. 93, pp. 55–92, 1991.

[3] 横内寛文. プログラム意味論, 情報数学講座, 第 7 巻. 共立出版, 1994.